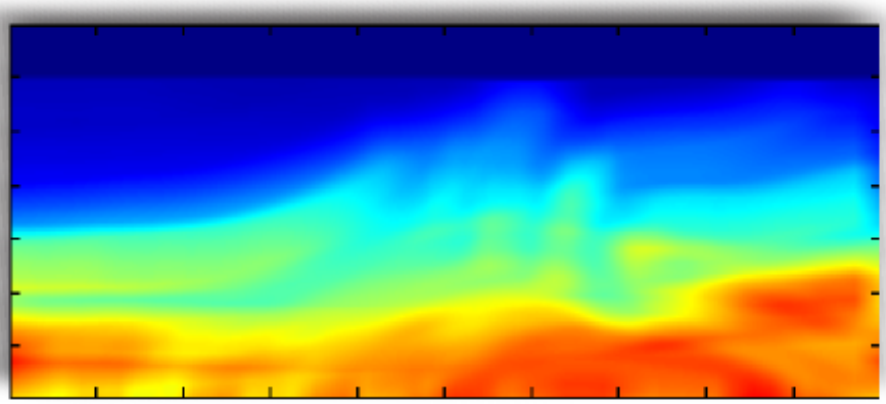
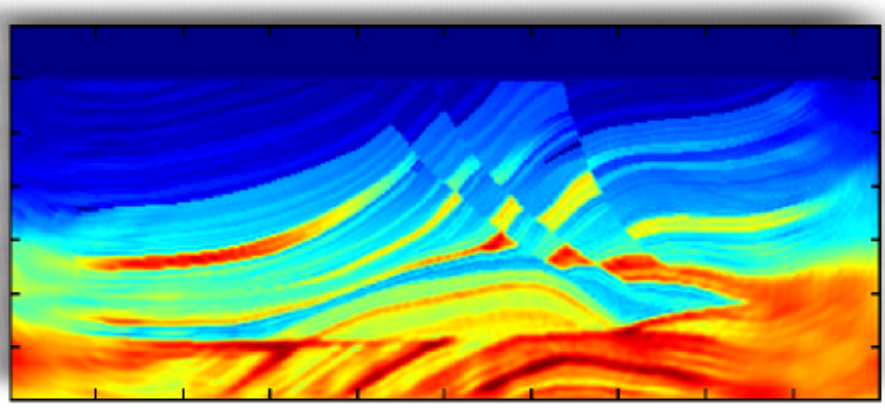
Christian-Albrechts-Universität (Kiel)
Karlsruher Institut für Technologie (Karlsruhe)

DENISE

User Manual





Daniel Köhn
Denise De Nil
André Kurzmann
Lisa Groos
Martin Schäfer
Sven Heider

DENISE
User Manual

© Christian-Albrechts-Universität Kiel (Germany) and
Karlsruher Institut für Technologie (KIT) (Germany)
Version 1.0

February 25, 2013

Authors

The DENISE code was first developed by Daniel Köhn, Denise De Nil and André Kurzmann at the Christian-Albrechts-Universität Kiel and TU Bergakademie Freiberg (Germany) from 2005 to 2009.

The forward code is based on the viscoelastic FD code fdveps (now SOFI2D) by Bohlen [2002].

Different external libraries for timedomain filtering are used.
The copyright of the source codes are held by different persons:

cseife.c, cseife.h, lib_stfinv, lib_aff, lib_fourier:
Copyright (c) 2005 by Thomas Forbriger (BFO Schiltach)

cseife_deriv.c, cseife_gauss.c, cseife_rekfl.c, cseife_rfk.c and cseife_tides.c:
Copyright (c) 1984 by Erhard Wielandt
This algorithm was part of seife.f. A current version of seife.f can be obtained from http://www.software-for-seismometry.de/

The Matlab implementation of a few SU routines, mainly used to read and write SU files are:
Copyright (C) 2008, Signal Analysis and Imaging Group
For more information: http://www-geo.phys.ualberta.ca/saig/SeismicLab
Author: M.D.Sacchi

Since then it has been developed and maintained by a development team: in alphabetical order,

Sven Heider,
Lisa Groos,
Martin Schäfer ...
(add other developers here in the future).

Contents

1 Introduction **4**
- 1.1 Citation . . . 5
- 1.2 Support . . . 5

2 Theoretical Background **7**
- 2.1 Equations of motion for an elastic medium . . . 7
- 2.2 Solution of the elastic wave equation by finite differences . . . 8
  - 2.2.1 Discretization of the wave equation . . . 8
  - 2.2.2 Accuracy of FD operators . . . 9
  - 2.2.3 Initial and Boundary Conditions . . . 10
- 2.3 Numerical Artefacts and Instabilities . . . 13
  - 2.3.1 Grid Dispersion . . . 13
  - 2.3.2 The Courant Instability . . . 16

3 The adjoint problem **18**
- 3.1 What is an "optimum" model ? . . . 18
- 3.2 How to find an optimum model . . . 19
- 3.3 Calculation of the gradient direction $\frac{\partial \mathrm{E}}{\partial \mathbf{m}}$ . . . 20
- 3.4 Gradients for the stress-velocity code . . . 26
- 3.5 Gradients for different model parametrizations . . . 26
- 3.6 Estimation of an optimum step length μ_{n} . . . 27
- 3.7 Nonlinear Conjugate Gradient Method . . . 29
- 3.8 The elastic FWT algorithm . . . 32

4 Source Time Wavelet Inversion **33**

5 Getting Started **36**
- 5.1 Requirements . . . 36
  - 5.1.1 LAM . . . 36
  - 5.1.2 How to run DENISE on the Linuxcluster at RZ Kiel . . . 37
- 5.2 Installation . . . 39
- 5.3 Compilation of DENISE . . . 39
- 5.4 Running the program . . . 41
- 5.5 Postprocessing . . . 43

6 Parameter definition with json input file **44**
- 6.1 Domain decomposition . . . 46
- 6.2 Discretization . . . 47
- 6.3 Order of the FD operator . . . 48
- 6.4 Time stepping . . . 48
- 6.5 Sources . . . 48
- 6.6 Model input . . . 51
- 6.7 Free surface . . . 51

6.8 Boundary conditions . . . 52
6.9 Receivers . . . 52
6.10 Seismograms . . . 53
6.11 Q-approximation . . . 54
6.12 Wavefield snapshots . . . 54
6.13 Receiver array . . . 55
6.14 Monitoring the simulation . . . 55
6.15 Checkpointing . . . 55
6.16 General inversion parameters . . . 56
6.17 Output of inversion results . . . 57
6.18 Change optimization method . . . 57
6.19 Misfit definition . . . 58
6.20 Step length estimation . . . 59
6.21 Abort criterion . . . 59
6.22 Minimum number of iteration per frequency . . . 60
6.23 Definition of the gradient taper geometry . . . 60
6.24 Definition of spatial filtering of the gradients . . . 61
6.25 Smoothing the gradients . . . 61
6.26 Limits for the model parameters . . . 61
6.27 Time windowing and damping . . . 62
6.28 Source wavelet inversion . . . 62
6.29 Smoothing the models . . . 63
6.30 Frequency filtering . . . 63
6.31 Trace killing . . . 64

7 Example 2 - the elastic Marmousi2 model 65
7.1 The complex Marmousi2 model . . . 65
7.1.1 Acquisition geometry and FD model . . . 68
7.1.2 Elastic wave propagation in the complex Marmousi model . . . 68
7.1.3 FWT of the complex Marmousi model . . . 70

8 Example 3 - inversion of ultrasonic data 78
8.1 Homogenous plexiglas bloc model . . . 78

9 Example 4 - shallow seismics 91
9.1 Inversion of viscoelastic observations . . . 91

Chapter 1

Introduction

The aim of Full Waveform Tomography (FWT) is to estimate the elastic material parameters in the underground. This can be achieved by minimizing the misfit energy between the modelled and field data using a gradient optimization approach. Because the FWT uses the full information content of each seismogram, structures below the seismic wavelength can be resolved. This is a tremendous improvement in resolution compared to traveltime tomography (Pratt et al. [2002]).
The concept of full waveform tomography was originally developed by Albert Tarantola in the 1980s for the acoustic, isotropic elastic, and viscoelastic case (Tarantola [1984b,a, 1986, 1988]). First numerical implementations were realized at the end of the 1980s (Gauthier et al. [1986], Mora [1987], Pica et al. [1990]), but due to limited computational resources, the application was restricted to simple 2D synthetic test problems and small near offset datasets. At the begining of the 1990s the original time domain formulation was transfered to a robust frequency domain approach (Pratt and Worthington [1990], Pratt [1990]). With the increasing performance of supercomputers moderately sized problems could be inverted with frequency domain approaches.
A spectacular result to prove the application of acoustic FWT on laboratory scale was presented by Pratt [1999] for ultrasonic tomography measurements on a simple block model. In a numerical blind test Brenders and Pratt [2007] achieved a very good agreement between their inversion result and the unkown true P-wave velocity model. The parallelization and performance optimizations of the frequency domain approach (see e.g. Sourbier et al. [2009a], Sourbier et al. [2009b]) lead to a wide range of acoustic FWT applications for problems on different scales, from the global scale, crustal scale over engineering and near surface scale, down to laboratory scale (Pratt [2004]).
Beside the application to geophysical problems, the acoustic FWT is also used to improve the resolution in medical cancer diagnostics (Pratt et al. [2007]). However, all these examples are restricted to the inversion of the acoustic material parameters: P-wave velocity, density and additionally the viscoacoustic damping Q_p for the P-waves. Even today the independent 2D FWT of all three isotropic elastic material parameters is still a challenge. Most elastic approaches invert for P-wave velocity only and use empirical relationships to deduce the distribution of S-wave velocity and density (Shipp and Singh [2002], Sheen et al. [2006]). Recently some authors also investigated the independent multiparameter FWT in the frequency domain (Choi et al. [2008a,b], Brossier [2009]).

In order to extract information about the structure and composition of the crust from seismic observations, it is necessary to be able to predict how seismic wavefields are affected by complex structures. Since exact analytical solutions to the wave equations do not exist for most subsurface configurations, the solutions can be obtained only by numerical methods. For iterative calculations of synthetic seismograms with limited computer resources fast and accurate modeling methods are needed.

The FD modeling/inversion program DENISE (subwavelength **DE**tail resolving **N**onlinear **I**terative **SE**ismic inversion), is based on the FD approach described by Virieux [1986] and Levander [1988]. The present program DENISE has the following extensions

- is efficently parallelized using domain decomposition with MPI Bohlen [2002],
- considers viscoelastic wave propagation effects like attenuation and dispersion Robertsson et al. [1994], Blanch et al. [1995], Bohlen [2002],
- employs higher order FD operators,

- applies Convolutional Perfectly Matched Layer boundary conditions at the edges of the numerical mesh Komatitsch and Martin [2007].

In the following sections, we give an extensive description of the theoretical background, the different input parameters and show a few benchmark modeling and inversion applications.

1.1 Citation

If you use this code for your own research, please cite at least one article written by the developers of the package, for instance:

Köhn [2011]

or

(XX add more references here)

and/or other articles from (`http://www.geophysik.uni-kiel.de/~dkoehn/publications.htm`)

The corresponding BibTEX entries may be found in file `doc/USER_MANUAL/thesis.bib`.

1.2 Support

The development of the code was suppported by the Christian-Albrechts-Universitï¿½t Kiel, TU Bergakademie Freiberg, Deutsche Forschungsgemeinschaft (DFG), Bundesministerium fï¿½r Bildung und Forschung (BMBF), the Wave Inversion Technology (WIT) Consortium and the Verbundnetz-Gas AG (VNG). The code was tested and optimized at the computing centres of Kiel University, TU Bergakademie Freiberg, TU Chemnitz, TU Dresden, the Karlsruhe Institute of Technology (KIT) and the Hochleistungsrechenzentrum Nord (HLRN 1+2).

Acknowledgments and contact

We thank for constructive discussions and further code improvements:

Anna Przebindowska (Karlsruhe Institute of Technology),
Olaf Hellwig (TU Bergakademie Freiberg),
Dennis Wilken and Wolfgang Rabbel (Christian-Albrechts-Universität Kiel).

Please e-mail your feedback, questions, comments, and suggestions to
Daniel Köhn (`dkoehn-AT-geophysik.uni-kiel.de`).

Chapter 2

Theoretical Background

2.1 Equations of motion for an elastic medium

The propagation of waves in a general elastic medium can be described by a system of coupled linear partial differential equations. They consist of the equations of motion

$$\rho \frac{\partial \mathrm{v_i}}{\partial \mathrm{t}} = \frac{\partial \sigma_{\mathrm{ij}}}{\partial \mathrm{x_j}} + \mathrm{f_i} \tag{2.1}$$

which simply state that the momentum of the medium, the product of density ρ and the displacement velocity $\mathrm{v_i}$, can be changed by surface forces, described by the stress tensor σ_{ij} or body forces $\mathrm{f_i}$. These equations describe a general medium, like gas, fluid, solid or plasma. The material specific properties are introduced by additional equations which describe how the medium reacts when a certain force is applied. In the isotropic elastic case this can be described by a linear stress-strain relationship:

$$\begin{aligned} \sigma_{\mathrm{ij}} &= \lambda \theta \delta_{\mathrm{ij}} + 2\mu \epsilon_{\mathrm{ij}} \\ \epsilon_{\mathrm{ij}} &= \frac{1}{2}\left(\frac{\partial \mathrm{u_i}}{\partial \mathrm{x_j}} + \frac{\partial \mathrm{u_j}}{\partial \mathrm{x_i}} \right) \end{aligned} \tag{2.2}$$

where λ and μ are the Lamé parameters, ϵ_{ij} the strain tensor and $\mathrm{u_i}$ the displacement. Using $\mathrm{v_i} = \frac{\partial \mathrm{u_i}}{\partial \mathrm{t}}$, (2.1) and (2.2) can be transformed into a system of second order partial differential equations:

$$\begin{aligned} \rho \frac{\partial^2 \mathrm{u_i}}{\partial \mathrm{t}^2} &= \frac{\partial \sigma_{\mathrm{ij}}}{\partial \mathrm{x_j}} + \mathrm{f_i} \\ \sigma_{\mathrm{ij}} &= \lambda \theta \delta_{\mathrm{ij}} + 2\mu \epsilon_{\mathrm{ij}} \\ \epsilon_{\mathrm{ij}} &= \frac{1}{2}\left(\frac{\partial \mathrm{u_i}}{\partial \mathrm{x_j}} + \frac{\partial \mathrm{u_j}}{\partial \mathrm{x_i}} \right) \end{aligned} \tag{2.3}$$

This expression is called **Stress-Displacement** formulation. Another common form of the elastic equations of motion can be deduced by taking the time derivative of the stress-strain relationship and the strain tensor in Eq. (2.3). Since the Lamé parameters λ and μ do not depend on time, Eq. (2.3) can be written as:

$$\begin{aligned} \rho \frac{\partial \mathrm{v_i}}{\partial \mathrm{t}} &= \frac{\partial \sigma_{\mathrm{ij}}}{\partial \mathrm{x_j}} + \mathrm{f_i} \\ \frac{\partial \sigma_{\mathrm{ij}}}{\partial \mathrm{t}} &= \lambda \frac{\partial \theta}{\partial \mathrm{t}} \delta_{\mathrm{ij}} + 2\mu \frac{\partial \epsilon_{\mathrm{ij}}}{\partial \mathrm{t}} \\ \frac{\partial \epsilon_{\mathrm{ij}}}{\partial \mathrm{t}} &= \frac{1}{2}\left(\frac{\partial \mathrm{v_i}}{\partial \mathrm{x_j}} + \frac{\partial \mathrm{v_j}}{\partial \mathrm{x_i}} \right) \end{aligned} \tag{2.4}$$

This expression is called **Stress-Velocity** formulation. For simple cases (2.3) and (2.4) can be solved analytically. More complex problems require numerical solutions. One possible approach for a numerical solution is described in the next section.

2.2 Solution of the elastic wave equation by finite differences

2.2.1 Discretization of the wave equation

For the numerical solution of the elastic equations of motion, Eqs. (2.3) have to be discretized in time and space on a grid. The particle displacement **u**, the stresses σ_{ij}, the Lamé parameters λ and μ are calculated and defined at discrete Cartesian coordinates $x = i\,dh$, $y = j\,dh$ and discrete times $t = n\,dt$. dh denotes the spatial distance between two adjacent grid points and dt the difference between two successive time steps. Therefore every grid point is located in the interval $i \in N|[1, NX]$, $j \in N|[1, NY]$ and $n \in N|[1, NT]$, where NX, NY and NT are the number of discrete spatial grid points and time steps, respectively. Finally the partial derivatives are replaced by **finite-difference (FD)** operators. Two types of operators can be distinguished, forward and backward operators D^+, D^-. The derivative of a function y with respect to a variable x can be approximated by the following operators:

$$\begin{aligned} D_x^+ y &= \frac{y[i+1] - y[i]}{dh} \qquad \text{forward operator} \\ D_x^- y &= \frac{y[i] - y[i-1]}{dh} \qquad \text{backward operator} \end{aligned} \tag{2.5}$$

To calculate the spatial derivatives of the wavefield variables at the correct positions, the variables are not placed on the same grid points, but staggered by half of the spatial grid point distance (Virieux [1986] and Levander [1988]). figure 2.1 shows the distribution of the material parameters and wavefield variables on the spatial grid.

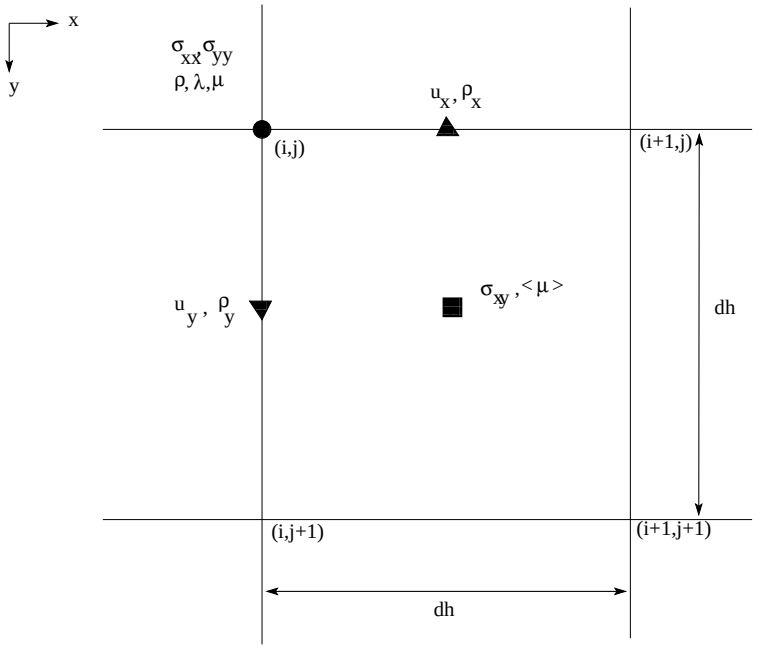


Figure 2.1: Grid geometry for a standard staggered grid (SSG) in Cartesian coordinates as suggested by Virieux [1986] and Levander [1988].

To guarantee the stability of the **standard staggered grid (SSG)** code, the Lamé parameter μ and density ρ have to be averaged harmonically and arithmetically (Moczo et al. [2004], Bohlen and Saenger [2006]), respectively

$$\begin{aligned} <\mu>[j+\frac{1}{2}][i+\frac{1}{2}] &= \left[\frac{1}{4}\left(\mu^{-1}[j][i] + \mu^{-1}[j][i+1] + \mu^{-1}[j+1][i+1] + \mu^{-1}[j+1][i]\right)\right]^{-1} \\ \rho_x[j][i+\frac{1}{2}] &= \frac{1}{2}(\rho[j][i+1] + \rho[j][i]) \\ \rho_y[j+\frac{1}{2}][i] &= \frac{1}{2}(\rho[j+1][i] + \rho[j][i]) \end{aligned} \tag{2.6}$$

The discretization of the linear stress strain relationship in (2.3) at time step n leads to the following system of equations (for simplicity I skip the time index n):

$$\begin{aligned}
\mathrm{u_{xx}[j][i]} &\approx \frac{\mathrm{u_x[j][i+\frac{1}{2}]} - \mathrm{u_x[j][i-\frac{1}{2}]}}{\mathrm{dh}} \\
\mathrm{u_{yy}[j][i]} &\approx \frac{\mathrm{u_y[j+\frac{1}{2}][i]} - \mathrm{u_y[j-\frac{1}{2}][i]}}{\mathrm{dh}} \\
\mathrm{u_{yx}[j+\frac{1}{2}][i+\frac{1}{2}]} &\approx \frac{\mathrm{u_y[j+\frac{1}{2}][i+1]} - \mathrm{u_y[j+\frac{1}{2}][i]}}{\mathrm{dh}} \\
\mathrm{u_{xy}[j+\frac{1}{2}][i+\frac{1}{2}]} &\approx \frac{\mathrm{u_x[j+1][i+\frac{1}{2}]} - \mathrm{u_x[j][i+\frac{1}{2}]}}{\mathrm{dh}} \\
\sigma_{\mathrm{xy}}\mathrm{[j+\frac{1}{2}][i+\frac{1}{2}]} &= <\mu> \mathrm{[j+\frac{1}{2}][i+\frac{1}{2}]} \left(\mathrm{u_{xy}[j+\frac{1}{2}][i+\frac{1}{2}]} + \mathrm{u_{yx}[j+\frac{1}{2}][i+\frac{1}{2}]} \right) \\
\sigma_{\mathrm{xx}}\mathrm{[j][i]} &= \lambda\mathrm{[j][i]} * \left(\mathrm{u_{xx}[j][i]} + \mathrm{u_{yy}[j][i]} \right) + 2 * <\mu> \mathrm{[j][i]} * \mathrm{u_{xx}[j][i]} \\
\sigma_{\mathrm{yy}}\mathrm{[j][i]} &= \lambda\mathrm{[j][i]} * \left(\mathrm{u_{xx}[j][i]} + \mathrm{u_{yy}[j][i]} \right) + 2 * <\mu> \mathrm{[j][i]} * \mathrm{u_{yy}[j][i]}
\end{aligned} \tag{2.7}$$

The discretization of the momentum equation in (2.3) leads to the following system of equations:

$$\begin{aligned}
\mathrm{utt_x^n[j][i+\frac{1}{2}]} &= \left(\sigma_{\mathrm{xx}}\mathrm{[j][i+1]} - \sigma_{\mathrm{xx}}\mathrm{[j][i]} + \sigma_{\mathrm{xy}}\mathrm{[j+\frac{1}{2}][i]} - \sigma_{\mathrm{xy}}\mathrm{[j-\frac{1}{2}][i]} \right) \\
\mathrm{utt_y^n[j+\frac{1}{2}][i]} &= \left(\sigma_{\mathrm{xy}}\mathrm{[j][i+\frac{1}{2}]} - \sigma_{\mathrm{xy}}\mathrm{[j][i-\frac{1}{2}]} + \sigma_{\mathrm{yy}}\mathrm{[j+1][i]} - \sigma_{\mathrm{yy}}\mathrm{[j][i]} \right) \\
\mathrm{u_x^{n+1}[j][i+\frac{1}{2}]} &= 2 * \mathrm{u_x^n[j][i+\frac{1}{2}]} - \mathrm{u_x^{n-1}[j][i+\frac{1}{2}]} + \frac{\mathrm{dt}^2}{\mathrm{dh} * \rho_{\mathrm{x}}\mathrm{[j][i+\frac{1}{2}]}} * \mathrm{utt_x^n[j][i+\frac{1}{2}]} \\
\mathrm{u_y^{n+1}[j+\frac{1}{2}][i]} &= 2 * \mathrm{u_y^n[j+\frac{1}{2}][i]} - \mathrm{u_y^{n-1}[j+\frac{1}{2}][i]} + \frac{\mathrm{dt}^2}{\mathrm{dh} * \rho_{\mathrm{y}}\mathrm{[j+\frac{1}{2}][i]}} * \mathrm{utt_y^n[j+\frac{1}{2}][i]}
\end{aligned} \tag{2.8}$$

2.2.2 Accuracy of FD operators

The derivation of the FD operators in the last section was a simple replacement of the partial derivatives by finite differences. In the following more systematic approach, the first derivative of a variable f at a grid point i is estimated by a Taylor series expansion (Jastram [1992]):

$$\begin{aligned}
(2\mathrm{k}-1)\left.\frac{\partial \mathrm{f}}{\partial \mathrm{x}}\right|_{\mathrm{i}} &= \frac{1}{\mathrm{dh}}(\mathrm{f_{i+(k-1/2)}} - \mathrm{f_{i-(k-1/2)}}) \\
&\quad + \frac{1}{\mathrm{dh}} \sum_{\mathrm{l}=2}^{\mathrm{N}} \frac{((\mathrm{k}-\frac{1}{2})\mathrm{dh})^{2\mathrm{l}-1}}{(2\mathrm{l}-1)!} \left.\frac{\partial^{(2\mathrm{l}-1)}\mathrm{f}}{\partial \mathrm{x}^{(2\mathrm{l}-1)}}\right|_{\mathrm{i}} + \mathcal{O}(\mathrm{dh})^{2\mathrm{N}}
\end{aligned}$$

For an operator with length 2N, N equations are added with a weight β_{k}:

$$\begin{aligned}
[\sum_{\mathrm{k}=1}^{\mathrm{N}} \beta_{\mathrm{k}}(2\mathrm{k}-1)]\left.\frac{\partial \mathrm{f}}{\partial \mathrm{x}}\right|_{\mathrm{i}} &= \frac{1}{\mathrm{dh}} \sum_{\mathrm{k}=1}^{\mathrm{N}} \beta_{\mathrm{k}}(\mathrm{f_{i+(k-1/2)}} - \mathrm{f_{i-(k-1/2)}}) \\
&\quad + \frac{1}{\mathrm{dh}} \sum_{\mathrm{k}=1}^{\mathrm{N}} \sum_{\mathrm{l}=2}^{\mathrm{N}} \beta_{\mathrm{k}} \frac{((\mathrm{k}-\frac{1}{2})\mathrm{dh})^{2\mathrm{l}-1}}{(2\mathrm{l}-1)!} \left.\frac{\partial^{(2\mathrm{l}-1)}\mathrm{f}}{\partial \mathrm{x}^{(2\mathrm{l}-1)}}\right|_{\mathrm{i}} + \mathcal{O}(\mathrm{dh})^{2\mathrm{N}}
\end{aligned} \tag{2.9}$$

The case N=1 leads to the FD operator derived in the last section, which has a length of 2N=2. The Taylor series is truncated after the first term ($\mathcal{O}(\mathrm{dh})^2$). Therefore this operator is called **2nd order FD operator** which refers to the truncation error of the Taylor series and not to the order of the approximated derivative. To understand equation (2.9)

better, we estimate a **4th order FD operator**. This operator has the length 2N = 4 or N=2. The sums in Eq. (2.9) lead to:

$$\begin{aligned}(\beta_1 + 3\beta_2)\frac{\partial \mathrm{f}}{\partial \mathrm{x}}\bigg|_{\mathrm{i}} = \frac{1}{\mathrm{dh}}(\beta_1(\mathrm{f}_{\mathrm{i}+1/2} - \mathrm{f}_{\mathrm{i}-1/2}) + \beta_2(\mathrm{f}_{\mathrm{i}+3/2} - \mathrm{f}_{\mathrm{i}-3/2})) \\ + \frac{\mathrm{dh}^3}{\mathrm{dh}}\left[\beta_1\frac{1}{8\cdot 3!} + \beta_2\frac{27}{8\cdot 3!}\right]\frac{\partial^3 \mathrm{f}}{\partial \mathrm{x}^3}\bigg|_{\mathrm{i}}\end{aligned} \tag{2.10}$$

The weights β_{k} can be calculated by the following approach: The factor in front of the partial derivative on the LHS of Eq. (2.10) should equal 1, therefore

$$(\beta_1 + 3\beta_2) = 1.$$

The coefficients in front of $\frac{\partial^3 \mathrm{f}}{\partial \mathrm{x}^3}\Big|_{\mathrm{i}}$ on the RHS of Eq. (2.10) should vanish:

$$(\beta_1 + 27\beta_2) = 0.$$

The weights β_{k} can be estimated by solving the matrix equation:

$$\begin{pmatrix} 1 & 3 \\ 1 & 27 \end{pmatrix} \cdot \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

The resulting coefficients are $\beta_1 = 9/8$ and $\beta_2 = -1/24$. Therefore the 4th order backward- and forward operators are:

$$\begin{aligned}\frac{\partial \mathrm{f}}{\partial \mathrm{x}}\bigg|_{\mathrm{i}+1/2} &= \frac{1}{\mathrm{dh}}[\beta_1(\mathrm{f}_{\mathrm{i}+1} - \mathrm{f}_{\mathrm{i}}) + \beta_2(\mathrm{f}_{\mathrm{i}+2} - \mathrm{f}_{\mathrm{i}-1})] \qquad \text{forward operator} \\ \frac{\partial \mathrm{f}}{\partial \mathrm{x}}\bigg|_{\mathrm{i}-1/2} &= \frac{1}{\mathrm{dh}}[\beta_1(\mathrm{f}_{\mathrm{i}} - \mathrm{f}_{\mathrm{i}-1}) + \beta_2(\mathrm{f}_{\mathrm{i}+1} - \mathrm{f}_{\mathrm{i}-2})] \qquad \text{backward operator}\end{aligned} \tag{2.11}$$

The coefficients β_{i} in the FD operator are called **Taylor coefficients**. The accuracy of higher order FD operators can be improved by seeking for FD coefficients β_{k} that approximate the first derivative in a certain frequency range (Holberg [1987]). These numerically optimized coefficients are called **Holberg coefficients**.

2.2.3 Initial and Boundary Conditions

To find a unique solution of the problem, initial and boundary conditions have to be defined. The initial conditions for the elastic forward problem are:

$$\begin{aligned}\mathrm{u}_{\mathrm{i}}(\mathbf{x}, \mathrm{t}) &= 0 \\ \frac{\partial \mathrm{u}_{\mathrm{i}}(\mathbf{x}, \mathrm{t})}{\partial \mathrm{t}} &= 0\end{aligned} \tag{2.12}$$

for all $\mathrm{x} \in \mathrm{V}$ at $\mathrm{t} = 0$.
For the geophysical application two types of boundary conditions are very important:

1. Horizontal Free Surface: The interface between the elastic medium and air at the surface is very important when trying to model surface waves or multiple reflections in a marine environment. Since all stresses in the normal direction at this interface vanish

   $$\sigma_{\mathrm{xy}} = \sigma_{\mathrm{yy}} = 0.0 \tag{2.13}$$

   this boundary condition is called (stress) **free surface**. Two types of implementations are common. In the implicit defintion of the free surface, a small layer with the acoustic parameters of air ($\mathrm{V_p} = 300\ \mathrm{m/s}$, $\mathrm{V_s} = 0.0\ \mathrm{m/s}$, $\rho = 1.25\ \mathrm{kg/m^3}$) is placed on top of the model. One advantage of the implicit definition of the free surface is the easy implementation of topography on the FD grid, however to get accurate results for surface waves or multiples, this approach requires a fine spatial sampling of the FD grid near the free surface. An explicit free surface can be implemented by using the mirroring technique by Levander, which leads to stable and accurate

solutions for plain interfaces (Levander [1988], Robertsson et al. [1995]). If the planar free surface is located at grid point $j = h$, the stress at this point is set to zero and the stresses below the free surface are mirrored with an inverse sign:

$$\begin{aligned} \sigma_{\mathrm{yy}}(\mathrm{h},\mathrm{i}) &= 0 \\ \sigma_{\mathrm{yy}}(\mathrm{h}-1,\mathrm{i}) &= -\sigma_{\mathrm{yy}}(\mathrm{h}+1,\mathrm{i}) \\ \sigma_{\mathrm{xy}}(\mathrm{h}-\frac{1}{2},\mathrm{i}+\frac{1}{2}) &= -\sigma_{\mathrm{xy}}(\mathrm{h}+\frac{1}{2},\mathrm{i}+\frac{1}{2}) \\ \sigma_{\mathrm{xy}}(\mathrm{h}-\frac{3}{2},\mathrm{i}+\frac{1}{2}) &= -\sigma_{\mathrm{xy}}(\mathrm{h}+\frac{3}{2},\mathrm{i}+\frac{1}{2}) \end{aligned} \tag{2.14}$$

When updating the stress component $\sigma_{\mathrm{xx}} = (\lambda + 2\mu)\mathrm{u}_{\mathrm{xx}} + \lambda \mathrm{u}_{\mathrm{yy}}$ at the free surface, only horizontal particle displacements should be used because vertical derivatives over the free surface lead to instabilities (Levander [1988]). The vertical derivative of the y-displacement u_{yy} can be replaced by using the boundary condition at the free surface:

$$\begin{aligned} \sigma_{\mathrm{yy}} &= (\lambda + 2\mu)\mathrm{u}_{\mathrm{yy}} + \lambda \mathrm{u}_{\mathrm{xx}} = 0 \\ \mathrm{u}_{\mathrm{yy}} &= -\frac{\lambda}{(\lambda + 2\mu)}\mathrm{u}_{\mathrm{xx}} \end{aligned} \tag{2.15}$$

Therefore the stress σ_{xx} can be written as

$$\sigma_{\mathrm{xx}} = \frac{4(\lambda\mu + \mu^2)}{\lambda + 2\mu}\mathrm{u}_{\mathrm{xx}} \tag{2.16}$$

2. Absorbing Boundary Conditions: Due to limited computational resources, the FD grid has to be as small as possible. To model problems with an infinite extension in different directions, e.g. a full or half-space problem, an artificial absorbing boundary condition has to be applied. A very effective way to damp the waves near the boundaries are **Perfectly Matched Layers (PMLs)**. This can be achieved by a coordinate stretch of the wave equations in the frequency domain (Komatitsch and Martin [2007]). The coordinate stretch creates exponentially decaying plane wave solutions in the absorbing boundary frame. The PML's are only reflectionless if the exact wave equation is solved. As soon as the problem is discretized (for example using finite differences) you are solving an approximate wave equation and the analytical perfection of the PML is no longer valid. To overcome this shortcoming the wavefield is damped by the damping function

$$\mathrm{c} = -\mathrm{V}_{\mathrm{pml}} * \frac{\log(\alpha)}{\mathrm{L}} \tag{2.17}$$

where $\mathrm{V}_{\mathrm{pml}}$ denotes the typical P-wave velocity of the medium in the absorbing boundary frame, $\alpha = 1 \times 10^{-4}$ and L is the thickness of the absorbing boundary layer. A comparison between the exponential damping and the PML boundary is shown in Fig.2.2. The PMLs are damping the seismic waves by a factor 5-10 more effective than the absorbing boundary frame.

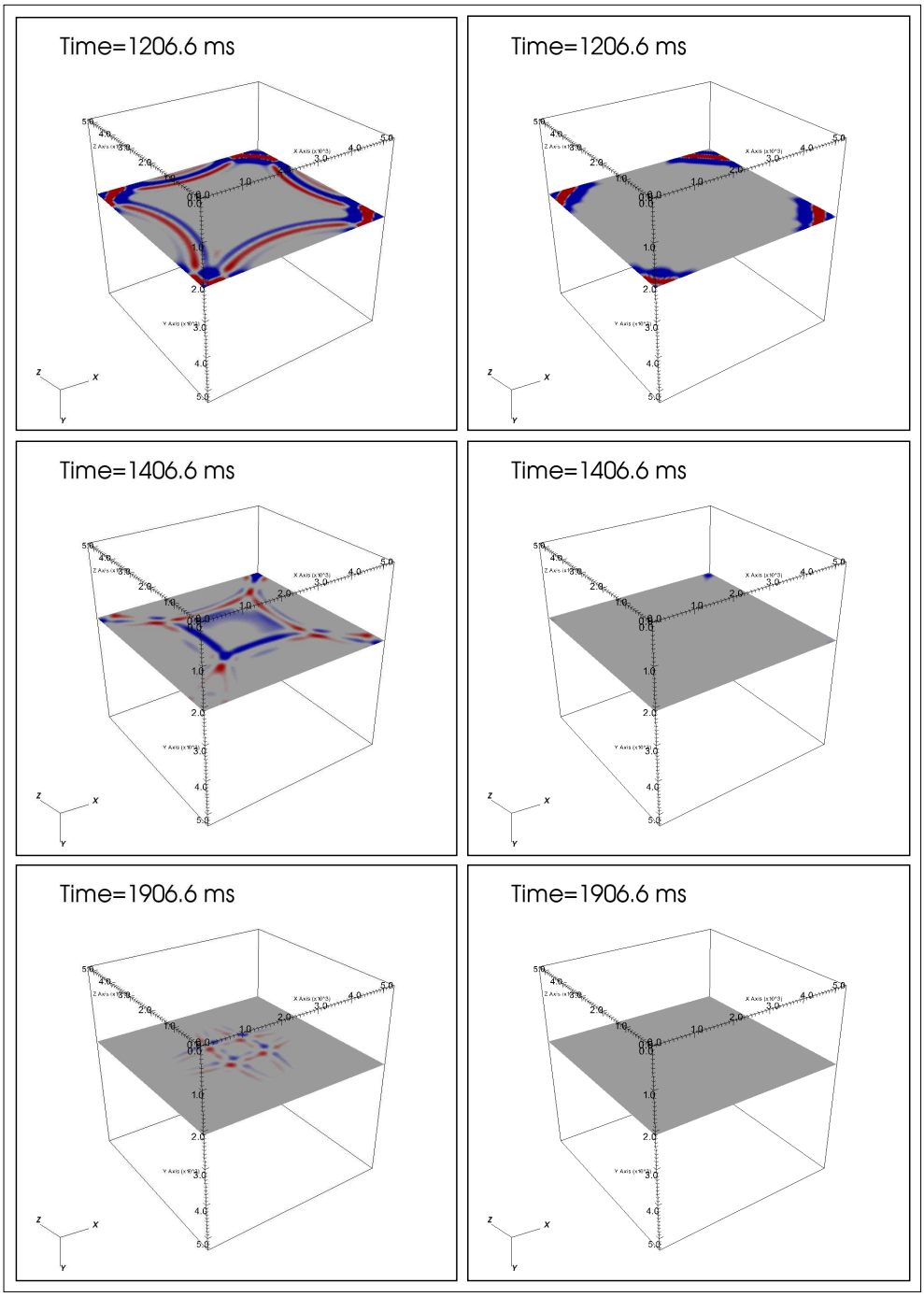


Figure 2.2: Comparison between exponential damping (left column) and PML (right column) absorbing boundary conditions for a homogeneous full space model.

2.3 Numerical Artefacts and Instabilities

To avoid numerical artefacts and instabilities during a FD modelling run, spatial and temporal sampling conditions for the wavefield have to be satisfied. These will be discussed in the following two sections.

2.3.1 Grid Dispersion

The first question when building a FD model is: What is the maximum spatial grid point distance dh, for a correct sampling of the wavefield ? To answer this question we take a look at this simple example: The particle displacement in x-direction is defined by a sine function:

$$\mathrm{u_x} = \sin\left(2\pi\frac{\mathrm{x}}{\lambda}\right), \tag{2.18}$$

where λ denotes the wavelength. When calculating the derivation of this function analytically at $\mathrm{x} = 0$ and setting $\lambda = 1\ \mathrm{m}$ we get:

$$\left.\frac{\mathrm{du_x}}{\mathrm{dx}}\right|_{\mathrm{x}=0} = \frac{2\pi}{\lambda}\cos\left(2\pi\frac{\mathrm{x}}{\lambda}\right)\Bigg|_{\mathrm{x}=0} = 2\pi. \tag{2.19}$$

In the next step the derivation is approximated numerically by a staggered 2nd order finite-difference operator:

$$\left.\frac{\mathrm{du_x}}{\mathrm{dx}}\right|_{\mathrm{x}=0} \approx \left.\frac{\mathrm{u_x}(\mathrm{x}+\frac{1}{2}\Delta\mathrm{x}) - \mathrm{u_x}(\mathrm{x}-\frac{1}{2}\Delta\mathrm{x})}{\Delta\mathrm{x}}\right|_{\mathrm{x}=0} = \frac{\sin\left(\frac{2\pi(\mathrm{x}+\frac{1}{2}\mathrm{dx})}{\lambda}\right) - \sin\left(\frac{2\pi(\mathrm{x}-\frac{1}{2}\mathrm{dx})}{\lambda}\right)}{\Delta\mathrm{x}}. \tag{2.20}$$

Using the Nyquist-Shannon sampling theorem it should be sufficient to sample the wavefield with $\Delta\mathrm{x} = \lambda/2$. In table 2.1 the numerical solutions of eq. (2.20) and the analytical solution (2.19) are compared for different sample intervals $\Delta\mathrm{x} = \lambda/\mathrm{n}$, where n is the number of gridpoints per wavelength. For the case n=2, which corresponds to the Nyquist-Shannon theorem, the numerical solution is $\frac{\mathrm{du_x}}{\mathrm{dx}}|_{\mathrm{x}=0} = 4.0$, which is not equal with the analytical solution 2π. A refinement of the spatial sampling of the wavefield results in an improvement of the finite difference solution. For $\mathrm{n} = 16$ the numerical solution is accurate to the second decimal place. The effect of a sparsly sampled pressure field is illustrated in figure 2.3 for a homogeneous block model with stress free surfaces. The dimensions of the FD grid are fixed and the central frequency of the source signal is increased systematically. When using a spatial sampling of 16 grid points per minimum wavelength (figure 2.3, top) the wavefronts are sharply defined. For $\mathrm{n} = 4$ grid points a slight numerical dispersion of the wave occurs (figure 2.3, center). This effect is obvious when using the Nyquist criterion ($\mathrm{n} = 2$) (figure 2.3, bottom). Since the numerical calculated wavefield seem to be dispersive this numerical artefact is called **grid dispersion**. To avoid the occurence of grid dispersion the following criteria for the spatial grid spacing dh has to be satisfied:

$$\mathrm{dh} \leq \frac{\lambda_{\mathrm{min}}}{\mathrm{n}} = \frac{\mathrm{V_{min}}}{\mathrm{n}\ \mathrm{f_{max}}}. \tag{2.21}$$

Here λ_{min} denotes the minimum wavelength, $\mathrm{V_{min}}$ the minimum velocity in the model and $\mathrm{f_{max}}$ is the maximum frequency of the source signal. Depending on the accuracy of the used FD operator the parameter n is different. In table 2.2 n is listed for different FD operator lengths and types (Taylor and Holberg operators). The Holberg coefficients are calculated for a minimum dispersion error of 0.1% at $3\mathrm{f_{max}}$. For short operators n should be choosen relatively large, so the spatial grid spacing is small, while for longer FD operators n is smaller and the grid spacing can be larger.

n	Δx [m]	$\frac{\mathrm{dv_x}}{\mathrm{dx}}\vert_{\mathrm{x}=0}$ []
analytical	-	$2\pi \approx 6.283$
2	$\lambda/2$	4.0
4	$\lambda/4$	5.657
8	$\lambda/8$	6.123
16	$\lambda/16$	6.2429
32	$\lambda/32$	6.2731

Table 2.1: Comparison of the analytical solution Eq. (2.19) with the numerical solution Eq. (2.20) for different grid spacings $\Delta \mathrm{x} = \lambda/\mathrm{n}$.

FDORDER	n (Taylor)	n (Holberg)
2nd	12	12
4th	8	8.32
6th	6	4.77
8th	5	3.69
10th	5	3.19
12th	4	2.91

Table 2.2: The number of grid points per minimum wavelength n for different orders (2nd-12th) and types (Taylor and Holberg) of FD operators. For the Holberg coefficients n is calculated for a minimum dispersion error of 0.1% at $3\mathrm{f}_{\mathrm{max}}$.

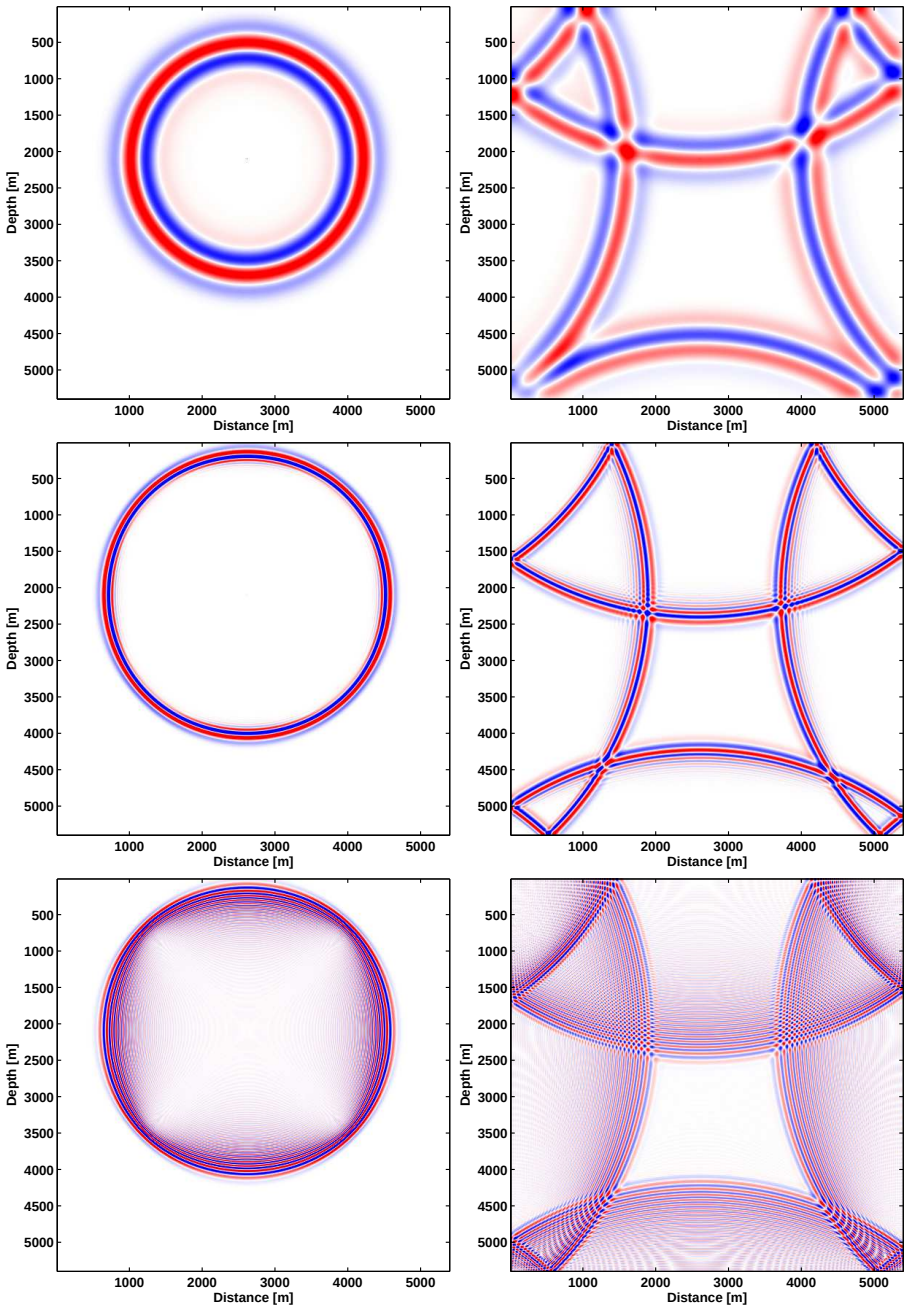


Figure 2.3: The influence of grid dispersion in FD modeling: Spatial sampling of the wavefield using n=16 (top), n=4 (center) and n=2 gridpoints (bottom) per minimum wavelength $\lambda_{\min}$.

2.3.2 The Courant Instability

Beside the spatial, the temporal discretization has to satisfy a sampling criterion to ensure the stability of the FD code. If a wave is propagating on a discrete grid, then the timestep dt has to be less than the time for the wave to travel between two adjacent grid points with grid spacing dh. For an elastic 2D grid this means mathematically:

$$\mathrm{dt} \leq \frac{\mathrm{dh}}{\mathrm{h}\sqrt{2}\mathrm{V}_{\max}}, \tag{2.22}$$

where $\mathrm{V}_{\max}$ is the maximum velocity in the model. The factor h depends on the order of the FD operator and can easily calculated by summing over the weighting coefficients β_{i}

$$\mathrm{h} = \sum_{\mathrm{i}} \beta_{\mathrm{i}}. \tag{2.23}$$

In table 2.3 h is listed for different FD operator lengths and types (Taylor and Holberg operators). Criterion (2.22) is called **Courant-Friedrichs-Lewy criterion** (Courant et al. [1928], Courant et al. [March 1967]). figure 2.4 shows the evolution of the pressure field when the Courant criterion is violated. After a few time steps the amplitudes are growing to infinity and the calculation becomes unstable.

FDORDER	h (Taylor)	h (Holberg)
2nd	1.0	1.0
4th	7/6	1.184614
6th	149/120	1.283482
8th	2161/1680	1.345927
10th	53089/40320	1.387660
12th	1187803/887040	1.417065

Table 2.3: The factor h in the Courant criterion for different orders (2nd-12th) and types (Taylor and Holberg) of FD operators.

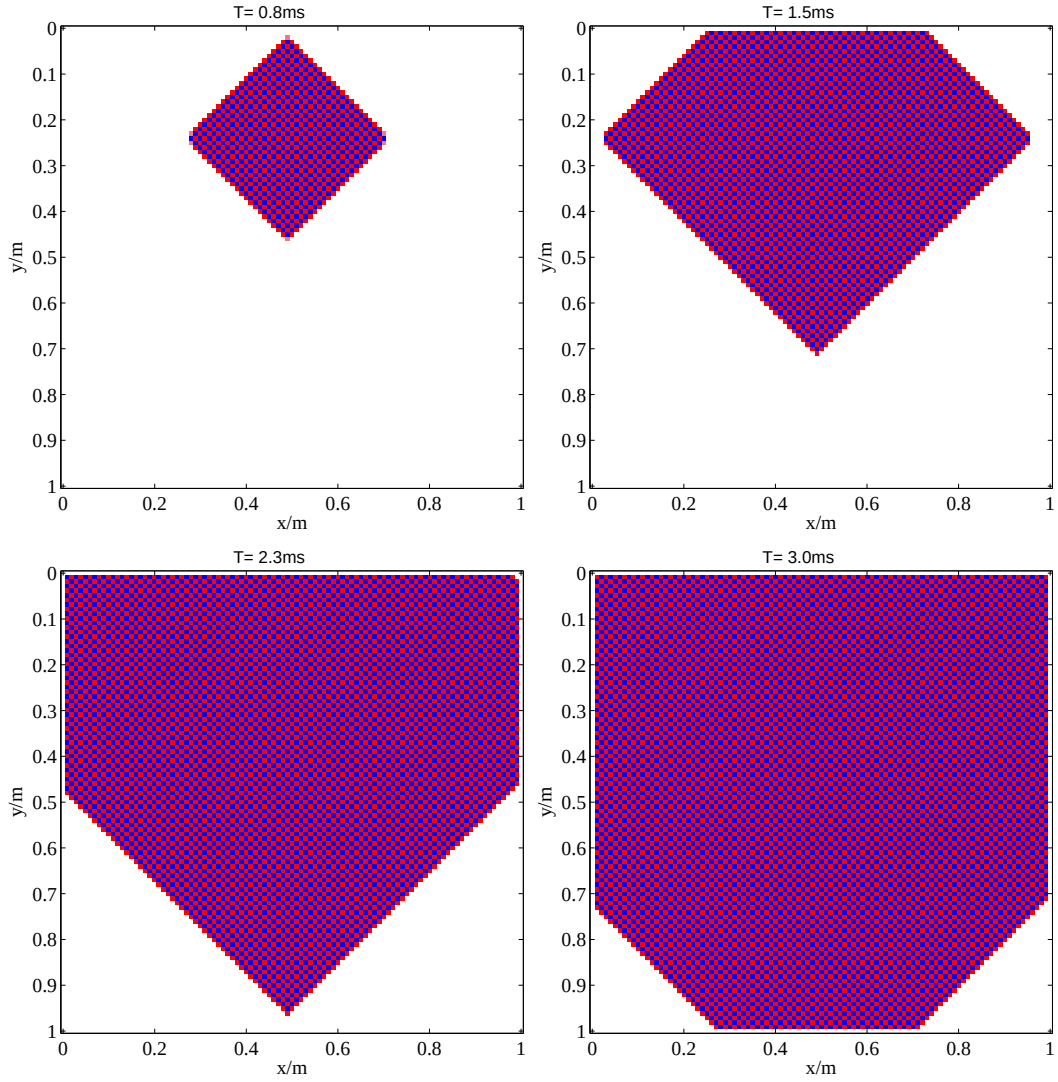


Figure 2.4: Temporal evolution of the Courant instability. In the colored areas the wave amplitudes are extremly large.

Chapter 3

The adjoint problem

The aim of full waveform tomography is to find an "optimum" model which can explain the data very well. It should not only explain the first arrivals of specific phases of the seismic wavefield like refractions or reflections, but also the amplitudes which contain information on the distribution of the elastic material parameters in the underground. To achieve this goal three problems have to be solved:

1. What is an "optimum" model ?
2. How can this model be found ?
3. Is this model unique or are other models existing, which could explain the data equally well ?

3.1 What is an "optimum" model ?

In reflection seismics the i^{th} component of the elastic displacement field $\mathrm{u_i}(\mathbf{x_s}, \mathbf{x_r}, \mathrm{t})$ excited by sources located at $\mathbf{x_s}$ will be recorded by receivers at $\mathbf{x_r}$ at time t. For a given distribution of the material parameters the forward problem Eq. 2.3 can be solved by finite differences (section 2.2). The result is a model data set $\mathbf{u^{mod}}$. This modelled data can be compared with the field data $\mathbf{u^{obs}}$. If the misfit or data residuals $\delta\mathbf{u} = \mathbf{u^{mod}} - \mathbf{u^{obs}}$ (figure 3.1) between the modelled and the field data is small the model can explain the data very well. If the residuals are large the model cannot explain the data. The misfit can be measured by a vector norm $|\mathrm{L}|_\mathrm{p}$ which is defined for $\mathrm{p} = 1, 2, ...$ as

$$|\mathrm{L}|_\mathrm{p} = \left(\sum_\mathrm{i} |\delta \mathrm{u_i}|^\mathrm{p} \right)^{1/\mathrm{p}} \tag{3.1}$$

The special case $|\mathrm{L}|_\infty$ is defined as

$$|\mathrm{L}|_\infty = \mathrm{max_i} |\delta \mathrm{u_i}|^\mathrm{p} \tag{3.2}$$

The L2-norm

$$\mathrm{E} = |\mathrm{L}|_2 = \frac{1}{2} \delta\mathbf{u^T} \delta\mathbf{u} \tag{3.3}$$

has a special physical meaning. It represents the residual elastic energy contained in the data residuals $\delta\mathbf{u}$. An optimum model can be found in a minimum of the residual energy. Therefore the optimum model is the solution of a nonlinear optimization problem.

3.2 How to find an optimum model

Figure 3.2 shows a schematic sketch of the residual energy at one point in space as a function of two model parameters λ and μ. The colors represent different values of the residual energy. Red areas represent models with high residual energy which do not fit the data, while the blue parts are good fitting models with low residual energies. The aim is to find the minimum of the residual energy marked by the red cross. Starting at a point $\mathbf{m_1} = (\lambda_1(\mathbf{x}), \mu_1(\mathbf{x}), \rho_1(\mathbf{x}),)$ in the parameter space we want to find the minimum by updating the material parameters in an iterative way

$$\mathbf{m}_2 = \mathbf{m}_1 + \mu_1 \delta \mathbf{m}_1, \tag{3.4}$$

along the search direction $\delta\mathbf{m}_1$ with the step length μ_1. To find the optimum search direction $\delta\mathbf{m}_1$ we expand the residual energy $\mathrm{E}(\mathbf{m}_1 + \delta\mathbf{m}_1)$ near the starting point in a Taylor series:

$$\mathrm{E}(\mathbf{m}_1 + \delta\mathbf{m}_1) \approx \mathrm{E}(\mathbf{m}_1) + \delta\mathbf{m}_1 \left(\frac{\partial \mathrm{E}}{\partial \mathrm{m}}\right)_1 + \frac{1}{2}\delta\mathbf{m}_1 \left(\frac{\partial^2 \mathrm{E}}{\partial \mathrm{m}^2}\right)_1 \delta\mathbf{m}_1^{\mathrm{T}} \tag{3.5}$$

and set the derivative of Eq. 3.5 with respect to $\delta\mathbf{m_1}$ zero

$$\frac{\partial \mathrm{E}(\mathbf{m}_1 + \delta\mathbf{m}_1)}{\partial \delta\mathbf{m}_1} = \left(\frac{\partial \mathrm{E}}{\partial \mathrm{m}}\right)_1 + \delta\mathbf{m}_1 \left(\frac{\partial^2 \mathrm{E}}{\partial \mathrm{m}^2}\right)_1 = 0 \tag{3.6}$$

Which finally leads to

$$\delta\mathbf{m}_1 = -\left(\frac{\partial^2 \mathrm{E}}{\partial \mathrm{m}^2}\right)_1^{-1} \left(\frac{\partial \mathrm{E}}{\partial \mathrm{m}}\right)_1 = -\mathbf{H_1}^{-1} \left(\frac{\partial \mathrm{E}}{\partial \mathrm{m}}\right)_1 \tag{3.7}$$

where $(\partial \mathrm{E}/\partial \mathbf{m})_1$ denotes the steepest-descent direction of the objective function and $\mathbf{H_1}^{-1}$ the inverse Hessian matrix. The inverse Hessian matrix for the elastic problem is often singular and can only be calculated with high computational costs. Therefore the inverse Hessian matrix is approximated by a preconditioning operator P. There is

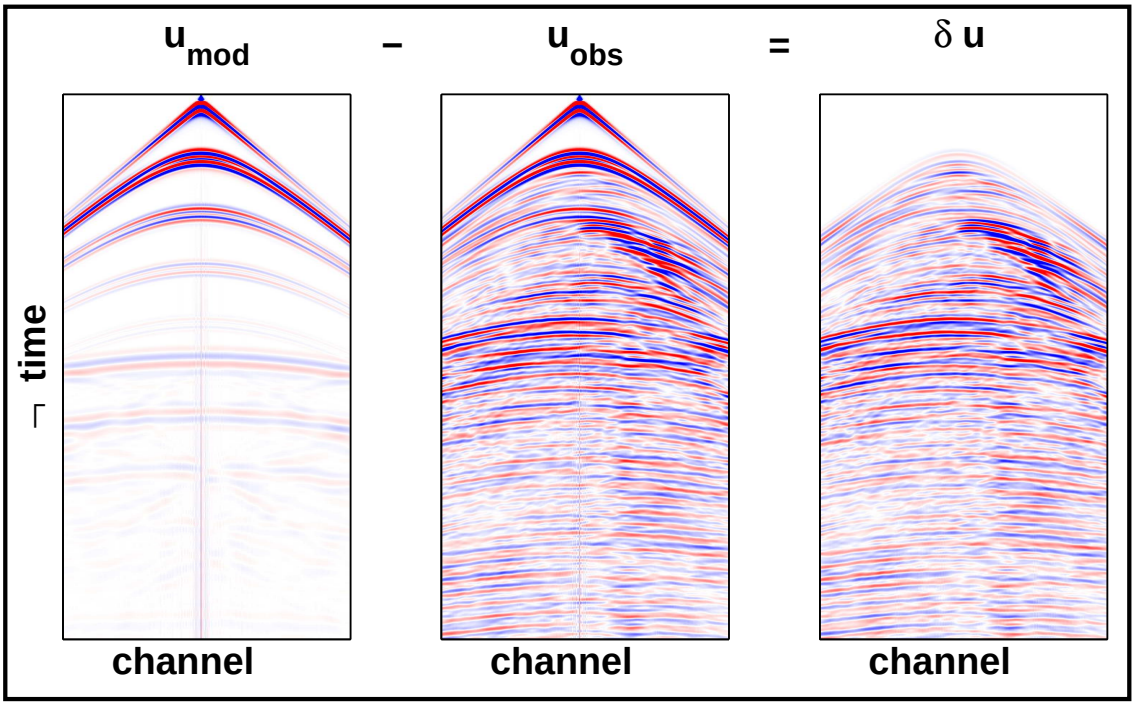


Figure 3.1: Definition of data residuals $\delta\mathbf{u}$.

no general rule for an optimum preconditioning operator.

$$\delta \mathbf{m}_1 \approx -\mathbf{P_1} \left(\frac{\partial \mathrm{E}}{\partial \mathrm{m}} \right)_1 . \tag{3.8}$$

By replacing $\delta \mathbf{m}_1$ in Eq. 3.4 with Eq. 3.8 we get

$$\mathbf{m}_2 = \mathbf{m}_1 - \mu_1 \mathbf{P_1} \left(\frac{\partial \mathrm{E}}{\partial \mathbf{m}} \right)_1 , \tag{3.9}$$

The optimum model parameters can be found along the negative gradient direction of the residual energy. The starting point $\mathbf{m_1}$ is not a particular point, so the update function can be applied to every point in the parameter space $\mathbf{m_n}$

$$\mathbf{m}_{\mathrm{n}+1} = \mathbf{m}_{\mathrm{n}} - \mu_{\mathrm{n}} \mathbf{P_n} \left(\frac{\partial \mathrm{E}}{\partial \mathbf{m}} \right)_{\mathrm{n}} . \tag{3.10}$$

3.3 Calculation of the gradient direction $\frac{\partial \mathrm{E}}{\partial \mathbf{m}}$

To estimate the gradient direction $\partial \mathrm{E} / \partial \mathbf{m}$ the residual energy is rewritten as:

$$\mathrm{E} = \frac{1}{2} \delta \mathbf{u}^{\mathbf{T}} \delta \mathbf{u} = \frac{1}{2} \sum_{\text{sources}} \int \mathrm{dt} \sum_{\text{receiver}} \delta \mathbf{u}^2 (\mathrm{x_r}, \mathrm{x_s}, \mathrm{t}) \tag{3.11}$$

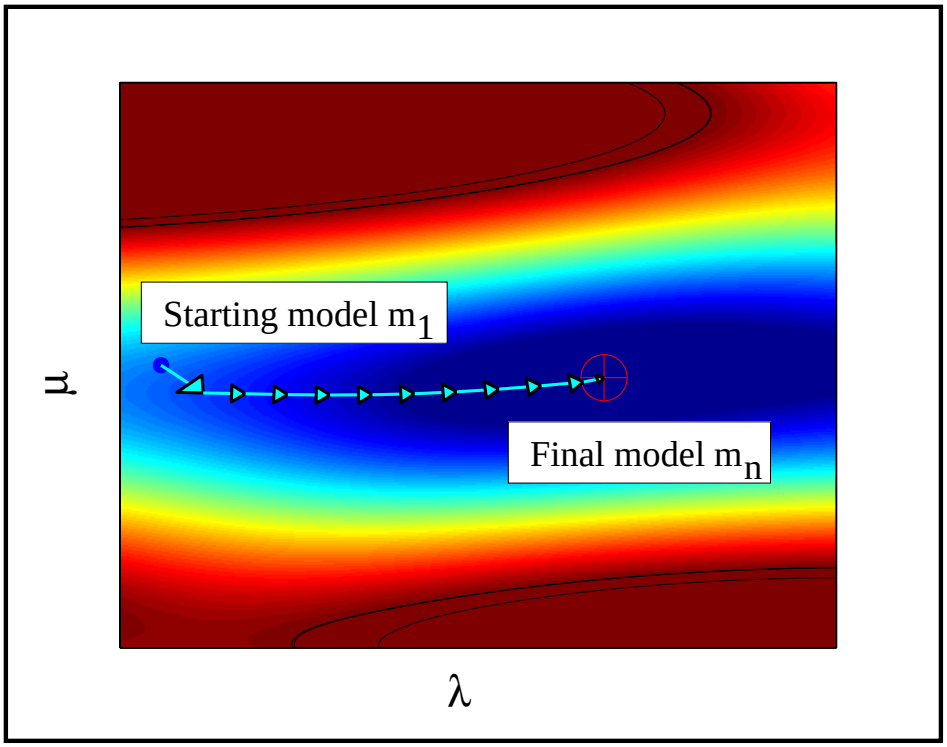


Figure 3.2: Schematic sketch of the residual energy at one point in space as a function of two model parameters m_1 and m_2. The blue dot denotes the starting point in the parameter space, while the red cross marks a minimum of the objective function.

After derivation with respect to a model parameter $\mathbf{m}$ we get

$$\begin{aligned}\frac{\partial \mathrm{E}}{\partial \mathbf{m}} &= \sum_{\text{sources}} \int \mathrm{dt} \sum_{\text{receiver}} \frac{\partial \delta \mathbf{u}}{\partial \mathbf{m}} \delta \mathbf{u} \\ &= \sum_{\text{sources}} \int \mathrm{dt} \sum_{\text{receiver}} \frac{\partial (\mathbf{u^{mod}}(\mathbf{m}) - \mathbf{u^{obs}})}{\partial \mathbf{m}} \delta \mathbf{u} \\ &= \sum_{\text{sources}} \int \mathrm{dt} \sum_{\text{receiver}} \frac{\partial \mathbf{u^{mod}}(\mathbf{m})}{\partial \mathbf{m}} \delta \mathbf{u}\end{aligned} \tag{3.12}$$

Eq. (3.12) can be related to the mapping of small changes from the data to the model space and vice versa (figure 3.3). A small change in the model space $\delta\mathbf{m}$, e.g. one model parameter at one point in space, will result in a small

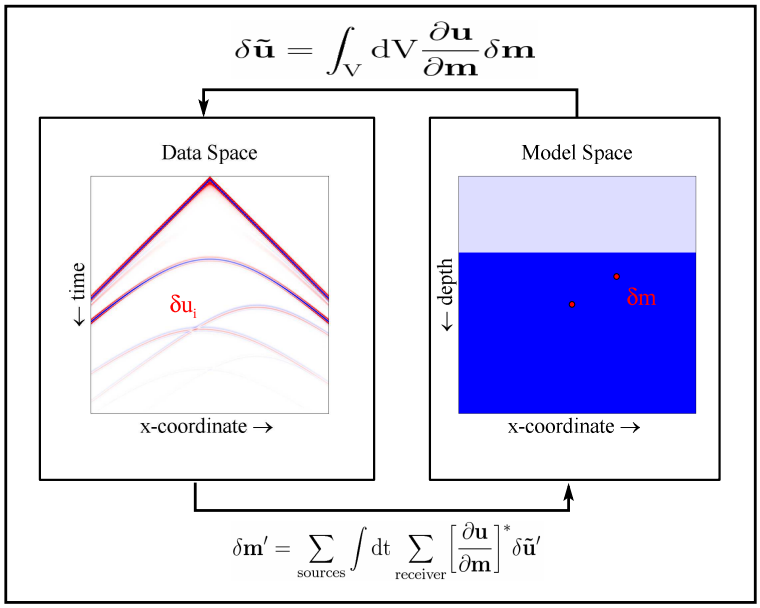


Figure 3.3: Mapping between model and data space and vice versa.

perturbation of the data space $\delta\tilde{\mathbf{u}}$, e.g. one wiggle in the seismic section. If the Frechét derivative $\frac{\partial \mathbf{u}}{\partial \mathbf{m}}$ is known, all the small perturbations in model space can be integrated over the model volume V to calculate the total change in data space (Tarantola [2005]):

$$\delta\tilde{\mathbf{u}}(\mathbf{x_s}, \mathbf{x_r}, \mathrm{t}) = \int_V \mathrm{dV} \frac{\partial \mathbf{u}}{\partial \mathbf{m}} \delta\mathbf{m}, \tag{3.13}$$

or by introducing the linear operator $\hat{\mathrm{L}}$

$$\delta\tilde{\mathbf{u}} = \hat{\mathrm{L}}\delta\mathbf{m} := \int_V \mathrm{dV} \frac{\partial \mathbf{u}}{\partial \mathbf{m}} \delta\mathbf{m}.$$

In a similar way small changes in the data space $\delta\tilde{\mathbf{u}}'$ can be integrated to calculate the total change in the model space $\delta\mathbf{m}'$ (Tarantola [2005])

$$\delta\mathbf{m}' = \sum_{\text{sources}} \int \mathrm{dt} \sum_{\text{receiver}} \left[\frac{\partial \mathbf{u}}{\partial \mathbf{m}}\right]^* \delta\tilde{\mathbf{u}}', \tag{3.14}$$

or as operator equation

$$\delta\mathbf{m}' = \hat{\mathrm{L}}^* \delta\tilde{\mathbf{u}}'.$$

In this case the Frechét derivative $\frac{\partial \mathbf{u}}{\partial \mathbf{m}}$ is replaced by it's adjoint counterpart $\frac{\partial \mathbf{u}}{\partial \mathbf{m}}^*$. Note that $\delta\tilde{\mathbf{u}} \neq \delta\tilde{\mathbf{u}}'$ and $\delta\mathbf{m} \neq \delta\mathbf{m}'$, so there is no unique way to map perturbations from the model to the data space or vice versa. Because the operator $\hat{\mathrm{L}}$ is linear, the kernel of $\hat{\mathrm{L}}$ and it's adjoint counterpart $\hat{\mathrm{L}}^*$ are identical (see chapter 5.4.2 in Tarantola [2005])

$$\left[\frac{\partial \mathbf{u}}{\partial \mathbf{m}}\right]^* = \left[\frac{\partial \mathbf{u}}{\partial \mathbf{m}}\right]$$

Therefore the mapping from the data to the model space Eq. (3.14) is equal to the gradient of the residual energy Eq. (3.12):

$$\begin{aligned}\delta\mathbf{m}' &= \sum_{\text{sources}} \int \mathrm{dt} \sum_{\text{receiver}} \left[\frac{\partial \mathbf{u_i}}{\partial \mathbf{m}}\right]^* \delta\tilde{\mathbf{u}}' \\ &= \sum_{\text{sources}} \int \mathrm{dt} \sum_{\text{receiver}} \left[\frac{\partial \mathbf{u_i}}{\partial \mathbf{m}}\right] \delta\mathbf{u} \\ &= \frac{\partial \mathrm{E}}{\partial \mathbf{m}}\end{aligned} \tag{3.15}$$

if the perturbation of the data space $\delta\tilde{\mathbf{u}}'$ is interpretated as data residuals $\delta\mathbf{u}$. So the approach to estimate the gradient direction $\partial \mathrm{E}/\partial \mathrm{m}$ can be split into 3 parts

1. Find a solution to the forward problem
$$\delta\mathbf{u} = \hat{\mathrm{L}}\delta\mathbf{m}.$$
2. Identify the Frechét kernels $\partial\mathbf{u}/\partial\mathbf{m}$
3. Use the property, that a linear operator $\hat{\mathrm{L}}$ and it's adjoint $\hat{\mathrm{L}}^*$ have the same kernels and calculate the gradient direction by using:
$$\frac{\partial \mathrm{E}}{\partial \mathbf{m}} = \delta\mathbf{m}' = \hat{\mathrm{L}}^*\delta\mathbf{u}'.$$

This is a very general approach. Now we apply this approach to the equations of motion for an elastic medium. The following derivation is much easier, when assuming a general elastic medium first and introduce the isotropy later on. Therefore the elastic forward problem Eqs. (2.3) can be written as

$$\begin{aligned}\rho\frac{\partial^2 \mathrm{u_i}}{\partial \mathrm{t}^2} - \frac{\partial}{\partial \mathrm{x_j}}\sigma_{\mathrm{ij}} &= \mathrm{f_i}, \\ \sigma_{\mathrm{ij}} - \mathrm{c_{ijkl}}\epsilon_{\mathrm{kl}} &= \mathrm{T_{ij}}, \\ \epsilon_{\mathrm{ij}} &= \frac{1}{2}\left(\frac{\partial \mathrm{u_i}}{\partial \mathrm{x_j}} + \frac{\partial \mathrm{u_j}}{\partial \mathrm{x_i}}\right), \\ &+ \text{initial and boundary conditions},\end{aligned} \tag{3.16}$$

where ρ denotes the density, σ_{ij} the stress tensor, ϵ_{ij} the strain tensor, c_{ijkl} the stiffness tensor, $\mathrm{f_i}$, $\mathrm{T_{ij}}$ source terms for volume and surface forces, respectively. In the next step every parameter and variable in the elastic wave equation is perturbated by a first order perturbation as shown in Fig. 3.3:

$$\begin{aligned}\mathrm{u_i} &\rightarrow \mathrm{u_i} + \delta\mathrm{u_i}, \\ \rho &\rightarrow \rho + \delta\rho, \\ \sigma_{\mathrm{ij}} &\rightarrow \sigma_{\mathrm{ij}} + \delta\sigma_{\mathrm{ij}}, \\ \mathrm{c_{ijkl}} &\rightarrow \mathrm{c_{ijkl}} + \delta\mathrm{c_{ijkl}}, \\ \epsilon_{\mathrm{ij}} &\rightarrow \epsilon_{\mathrm{ij}} + \delta\epsilon_{\mathrm{ij}},\end{aligned} \tag{3.17}$$

These substitutions yield new equations of motion describing the displacement perturbations δu_i and stress perturbations $\delta\sigma_{ij}$ as a function of new source terms Δf_i and ΔT_{ij}

$$\begin{aligned} \rho\frac{\partial^2\delta u_i}{\partial t^2} - \frac{\partial}{\partial x_j}\delta\sigma_{ij} &= \Delta f_i, \\ \delta\sigma_{ij} - c_{ijkl}\delta\epsilon_{kl} &= \Delta T_{ij}, \\ \delta\epsilon_{ij} &= \frac{1}{2}\left(\frac{\partial\delta u_i}{\partial x_j} + \frac{\partial\delta u_j}{\partial x_i}\right) \\ &\text{+ perturbated initial and boundary conditions} \end{aligned} \tag{3.18}$$

The new source terms are

$$\Delta f_i = -\delta\rho\frac{\partial^2 u_i}{\partial t^2} \tag{3.19}$$

and

$$\Delta T_{ij} = \delta c_{ijkl}\epsilon_{kl}. \tag{3.20}$$

Two points are important to notice:

1. Eq.(3.18) states that every change of a material parameter acts as a source (Eq.(3.19) and Eq.(3.20)), but the perturbated wavefield is propagating in the unperturbated medium.
2. The new wave equation (3.18) has mathematically the same form as the unperturbated elastic wave equation, and hence its solution can be obtained in terms of Green's functions G_{ij} of the elastic wave equation.

The solution of the perturbated elastic equations of motion (3.18) in terms of the elastic Green's function $G_{ij}(\mathbf{x}, t; \mathbf{x}', t')$ can be written as:

$$\begin{aligned} \delta u_i(\mathbf{x}, t) = &\int_V dV \int_0^T dt' G_{ij}(x, t; x', t')\Delta f_j(\mathbf{x}', t') \\ &- \int_V dV \int_0^T dt' \frac{\partial G_{ij}}{\partial x'_k}(x, t; x', t')\Delta T_{jk}(\mathbf{x}', t'). \end{aligned} \tag{3.21}$$

Substituting the force and traction terms given by Eqs.(3.19) and (3.20) into Eq.(3.21) yields after some rearranging

$$\begin{aligned} \delta u_i(\mathbf{x}, t) = &-\int_V dV \int_0^T dt' G_{ij}(x, t; x', t')\frac{\partial^2 u_j}{\partial t^2}(\mathbf{x}', t')\delta\rho \\ &- \int_V dV \int_0^T dt' \frac{\partial G_{ij}}{\partial x'_k}(x, t; x', t')\epsilon_{lm}(\mathbf{x}', t')\delta c_{jklm} \end{aligned} \tag{3.22}$$

Introducing isotropy leads to:

$$\begin{aligned} \delta u_i(\mathbf{x}, t) = &-\int_V dV \left[\int_0^T dt' G_{ij}(x, t; x', t')\frac{\partial^2 u_j}{\partial t^2}(\mathbf{x}', t')\right]\delta\rho \\ &- \int_V dV \left[\int_0^T dt' \frac{\partial G_{ij}}{\partial x'_k}(x, t; x', t')\epsilon_{lm}(\mathbf{x}', t')\delta_{jk}\delta_{lm}\right]\delta\lambda \\ &- \int_V dV \left[\int_0^T dt' \frac{\partial G_{ij}}{\partial x'_k}(x, t; x', t')\epsilon_{lm}(\mathbf{x}', t')(\delta_{jl}\delta_{lm} + \delta_{jm}\delta_{kl})\right]\delta\mu. \end{aligned} \tag{3.23}$$

Utilization of Eq.(3.23) to solve the forward problem is known as the Born approximation. In waveform tomography the Born approximation is not used to solve the forward problem. Instead the full elastic wave equation is solved. Equation (3.23) has the same form as the desired expression for the forward problem Eqs.(3.13):

$$\delta\mathbf{u} = \int_V dV \frac{\partial\mathbf{u}}{\partial\mathbf{m}}\delta\mathbf{m}. \tag{3.24}$$

Therefore the Frechét kernels $\frac{\partial \mathrm{u_i}}{\partial \mathbf{m}(\mathbf{x})}$ for the individual material parameters can be identified as:

$$\begin{aligned}
\frac{\partial \mathrm{u_i}}{\partial \rho} &= -\int_0^{\mathrm{T}} \mathrm{dt}' \mathrm{G_{ij}}(\mathrm{x},\mathrm{t};\mathrm{x}',\mathrm{t}') \frac{\partial^2 \mathrm{u_j}}{\partial \mathrm{t}^2}(\mathbf{x}',\mathrm{t}') \\
\frac{\partial \mathrm{u_i}}{\partial \lambda} &= -\int_0^{\mathrm{T}} \mathrm{dt}' \frac{\partial \mathrm{G_{ij}}}{\partial \mathrm{x}'_\mathrm{k}}(\mathrm{x},\mathrm{t};\mathrm{x}',\mathrm{t}') \epsilon_{\mathrm{lm}}(\mathbf{x}',\mathrm{t}') \delta_{\mathrm{jk}} \delta_{\mathrm{lm}} \\
\frac{\partial \mathrm{u_i}}{\partial \mu} &= -\int_0^{\mathrm{T}} \mathrm{dt}' \frac{\partial \mathrm{G_{ij}}}{\partial \mathrm{x}'_\mathrm{k}}(\mathrm{x},\mathrm{t};\mathrm{x}',\mathrm{t}') \epsilon_{\mathrm{lm}}(\mathbf{x}',\mathrm{t}') (\delta_{\mathrm{jl}} \delta_{\mathrm{lm}} + \delta_{\mathrm{jm}} \delta_{\mathrm{kl}})
\end{aligned} \tag{3.25}$$

By definition the adjoint of the operator (3.24) can be written as

$$\delta \mathbf{m}'(\mathbf{x}) = \sum_{\mathrm{sources}} \int_0^{\mathrm{T}} \mathrm{dt} \sum_{\alpha=1}^{\mathrm{N_{rec}}} \left[\frac{\partial \mathrm{u_i}}{\partial \mathbf{m}} \right]^* \delta \mathrm{u}'_\mathrm{i}(\mathbf{x}_\alpha, \mathrm{t}'), \tag{3.26}$$

Because a linear operator and its transpose have the same kernels $\partial \mathrm{u_i}/\partial \mathbf{m}$, the only difference arise in the variables of sum/integration, which are complementary. Inserting the integral kernels (3.25) in Eq.(3.26) yields

$$\begin{aligned}
\delta \rho' = &- \sum_{\mathrm{sources}} \int_0^{\mathrm{T}} \mathrm{dt} \sum_{\alpha=1}^{\mathrm{N_{rec}}} \int_0^{\mathrm{T}} \mathrm{dt}' \mathrm{G_{ij}}(\mathbf{x}_\alpha, \mathrm{t}'; \mathbf{x}, \mathrm{t}) \frac{\partial^2 \mathrm{u_j}}{\partial \mathrm{t}^2}(\mathbf{x}, \mathrm{t}) \delta \mathrm{u}'_\mathrm{i}(\mathrm{x}_\alpha, \mathrm{t}'), \\
\delta \lambda' = &- \sum_{\mathrm{sources}} \int_0^{\mathrm{T}} \mathrm{dt} \sum_{\alpha=1}^{\mathrm{N_{rec}}} \int_0^{\mathrm{T}} \mathrm{dt}' \frac{\partial \mathrm{G_{ij}}}{\partial \mathrm{x_k}}(\mathbf{x}_\alpha, \mathrm{t}'; \mathbf{x}, \mathrm{t}) \epsilon_{\mathrm{lm}}(\mathbf{x}, \mathrm{t}) \delta_{\mathrm{jk}} \delta_{\mathrm{lm}} \delta \mathrm{u}'_\mathrm{i}(\mathbf{x}_\alpha, \mathrm{t}'), \\
\delta \mu' = &- \sum_{\mathrm{sources}} \int_0^{\mathrm{T}} \mathrm{dt} \sum_{\alpha=1}^{\mathrm{N_{rec}}} \int_0^{\mathrm{T}} \mathrm{dt}' \frac{\partial \mathrm{G_{ij}}}{\partial \mathrm{x_k}}(\mathbf{x}_\alpha, \mathrm{t}'; \mathbf{x}, \mathrm{t}) \epsilon_{\mathrm{lm}}(\mathbf{x}, \mathrm{t}) (\delta_{\mathrm{jl}} \delta_{\mathrm{lm}} + \delta_{\mathrm{jm}} \delta_{\mathrm{kl}}) \delta \mathrm{u}'_\mathrm{i}(\mathbf{x}_\alpha, \mathrm{t}').
\end{aligned}$$

The terms only depending on time t and the positions $\mathbf{x}$ can be moved infront of the sum over the receivers

$$\begin{aligned}
\delta \rho' = &- \sum_{\mathrm{sources}} \int_0^{\mathrm{T}} \mathrm{dt} \frac{\partial^2 \mathrm{u_j}}{\partial \mathrm{t}^2}(\mathbf{x}, \mathrm{t}) \sum_{\alpha=1}^{\mathrm{N_{rec}}} \int_0^{\mathrm{T}} \mathrm{dt}' \mathrm{G_{ij}}(\mathbf{x}_\alpha, \mathrm{t}'; \mathbf{x}, \mathrm{t}) \delta \mathrm{u}'_\mathrm{i}(\mathrm{x}_\alpha, \mathrm{t}'), \\
\delta \lambda' = &- \sum_{\mathrm{sources}} \int_0^{\mathrm{T}} \mathrm{dt} \epsilon_{\mathrm{lm}}(\mathbf{x}, \mathrm{t}) \delta_{\mathrm{jk}} \delta_{\mathrm{lm}} \sum_{\alpha=1}^{\mathrm{N_{rec}}} \int_0^{\mathrm{T}} \mathrm{dt}' \frac{\partial \mathrm{G_{ij}}}{\partial \mathrm{x_k}}(\mathbf{x}_\alpha, \mathrm{t}'; \mathbf{x}, \mathrm{t}) \delta \mathrm{u}'_\mathrm{i}(\mathbf{x}_\alpha, \mathrm{t}'), \\
\delta \mu' = &- \sum_{\mathrm{sources}} \int_0^{\mathrm{T}} \mathrm{dt} \epsilon_{\mathrm{lm}}(\mathbf{x}, \mathrm{t}) (\delta_{\mathrm{jl}} \delta_{\mathrm{lm}} + \delta_{\mathrm{jm}} \delta_{\mathrm{kl}}) \sum_{\alpha=1}^{\mathrm{N_{rec}}} \int_0^{\mathrm{T}} \mathrm{dt}' \frac{\partial \mathrm{G_{ij}}}{\partial \mathrm{x_k}}(\mathbf{x}_\alpha, \mathrm{t}'; \mathbf{x}, \mathrm{t}) \delta \mathrm{u}'_\mathrm{i}(\mathbf{x}_\alpha, \mathrm{t}').
\end{aligned} \tag{3.27}$$

Defining the wavefield

$$\Psi_\mathrm{j}(\mathbf{x}, \mathrm{t}) = \sum_{\alpha=1}^{\mathrm{N_{rec}}} \int_0^{\mathrm{T}} \mathrm{dt}' \mathrm{G_{ij}}(\mathrm{x}_\alpha, \mathrm{t}'; \mathbf{x}, \mathrm{t}) \delta \mathrm{u}'_\mathrm{i}(\mathrm{x}_\alpha, \mathrm{t}'), \tag{3.28}$$

Eqs.(3.27) can be written as

$$\begin{aligned}
\delta \rho' = &- \sum_{\mathrm{sources}} \int_0^{\mathrm{T}} \mathrm{dt} \frac{\partial^2 \mathrm{u_j}}{\partial \mathrm{t}^2}(\mathbf{x}, \mathrm{t}) \Psi_\mathrm{j}, \\
\delta \lambda' = &- \sum_{\mathrm{sources}} \int_0^{\mathrm{T}} \mathrm{dt} \epsilon_{\mathrm{lm}}(\mathbf{x}, \mathrm{t}) \delta_{\mathrm{jk}} \delta_{\mathrm{lm}} \frac{\partial \Psi_\mathrm{j}}{\partial \mathrm{x_k}}, \\
\delta \mu' = &- \sum_{\mathrm{sources}} \int_0^{\mathrm{T}} \mathrm{dt} \epsilon_{\mathrm{lm}}(\mathbf{x}, \mathrm{t}) (\delta_{\mathrm{jl}} \delta_{\mathrm{lm}} + \delta_{\mathrm{jm}} \delta_{\mathrm{kl}}) \frac{\partial \Psi_\mathrm{j}}{\partial \mathrm{x_k}}.
\end{aligned} \tag{3.29}$$

The wavefield Ψ_j is generated by propagating the residual data $\delta u_i'$ from the receiver positions backwards in time through the elastic medium. To obtain a more symmetric expression for the density gradient, let us integrate the density gradient in (3.29) by parts

$$\begin{aligned}\delta\rho' &= -\sum_{\text{sources}}\int_0^T dt\left(\frac{\partial^2 u_j}{\partial t^2}(\mathbf{x},t)\right)\Psi_j(\mathbf{x},t)\\ &= -\sum_{\text{sources}}\left\{\left[\frac{\partial u_j}{\partial t}(\mathbf{x},T)\Psi_j(\mathbf{x},T)\right]_0^T - \int_0^T dt\frac{\partial u_j}{\partial t}(\mathbf{x},t)\frac{\partial \Psi_j}{\partial t}(\mathbf{x},t)\right\}.\end{aligned} \tag{3.30}$$

According to Eqs. (2.12) the field $u_j(\mathbf{x},t)$ satisfies initial conditions of rest, $u_j(\mathbf{x},0)=0$ and $\partial u_j(\mathbf{x},0)/\partial t = 0$. The field $\Psi_j(\mathbf{x},t)$ satisfies final conditions of rest, $\Psi_j(x,T)=0$. Therefore

$$\delta\rho' = -\sum_{\text{sources}}\int_0^T dt\left(\frac{\partial^2 u_j}{\partial t^2}(\mathbf{x},t)\right)\Psi_j(\mathbf{x},t) = \sum_{\text{sources}}\int_0^T dt\frac{\partial u_j}{\partial t}(\mathbf{x},t)\frac{\partial \Psi_j}{\partial t}(\mathbf{x},t). \tag{3.31}$$

Writing out the implicit sums in the gradients of the Lamé parameters $\delta\lambda'$ and $\delta\mu'$ in Eqs. (3.29)

$$\begin{aligned}\delta\lambda' &= -\sum_{\text{sources}}\int_0^T dt\sum_l\sum_k\sum_j\sum_m \epsilon_{lm}(\mathbf{x},t)\delta_{jk}\delta_{lm}\frac{\partial\Psi_j}{\partial x_k},\\ \delta\mu' &= -\sum_{\text{sources}}\int_0^T dt\sum_l\sum_k\sum_j\sum_m \epsilon_{lm}(\mathbf{x},t)(\delta_{jl}\delta_{lm}+\delta_{jm}\delta_{kl})\frac{\partial\Psi_j}{\partial x_k}.\end{aligned} \tag{3.32}$$

and neglecting all wavefield components and derivatives in z-direction leads to

$$\begin{aligned}\delta\lambda' &= -\sum_{\text{sources}}\int_0^T dt\left(\epsilon_{xx}+\epsilon_{yy}\right)\left(\frac{\partial\Psi_x}{\partial x}+\frac{\partial\Psi_y}{\partial y}\right),\\ \delta\mu' &= -\sum_{\text{sources}}\int_0^T dt\left[\left(\epsilon_{xy}+\epsilon_{yx}\right)\left(\frac{\partial\Psi_x}{\partial y}+\frac{\partial\Psi_y}{\partial x}\right)\right] + 2\left(\epsilon_{xx}\frac{\partial\Psi_x}{\partial x}+\epsilon_{yy}\frac{\partial\Psi_y}{\partial y}\right).\end{aligned} \tag{3.33}$$

Using the definition of the strain tensor ϵ_{ij} we get

$$\begin{aligned}\delta\lambda' &= -\sum_{\text{sources}}\int_0^T dt\left(\frac{\partial u_x}{\partial x}+\frac{\partial u_y}{\partial y}\right)\left(\frac{\partial\Psi_x}{\partial x}+\frac{\partial\Psi_y}{\partial y}\right),\\ \delta\mu' &= -\sum_{\text{sources}}\int_0^T dt\left[\left(\frac{\partial u_x}{\partial y}+\frac{\partial u_y}{\partial x}\right)\left(\frac{\partial\Psi_x}{\partial y}+\frac{\partial\Psi_y}{\partial x}\right)\right] + 2\left(\frac{\partial u_x}{\partial x}\frac{\partial\Psi_x}{\partial x}+\frac{\partial u_y}{\partial y}\frac{\partial\Psi_y}{\partial y}\right).\end{aligned} \tag{3.34}$$

Finally the gradients for the Lamé parameters λ, μ and the density ρ can be written as

$$\begin{aligned}\delta\lambda' &= -\sum_{\text{sources}}\int dt\left(\frac{\partial u_x}{\partial x}+\frac{\partial u_y}{\partial y}\right)\left(\frac{\partial\Psi_x}{\partial x}+\frac{\partial\Psi_y}{\partial y}\right)\\ \delta\mu' &= -\sum_{\text{sources}}\int dt\left(\frac{\partial u_x}{\partial y}+\frac{\partial u_y}{\partial x}\right)\left(\frac{\partial\Psi_x}{\partial y}+\frac{\partial\Psi_y}{\partial x}\right) + 2\left(\frac{\partial u_x}{\partial x}\frac{\partial\Psi_x}{\partial x}+\frac{\partial u_y}{\partial y}\frac{\partial\Psi_y}{\partial y}\right)\\ \delta\rho' &= \sum_{\text{sources}}\int dt\left(\frac{\partial u_x}{\partial t}\frac{\partial\Psi_x}{\partial t}+\frac{\partial u_y}{\partial t}\frac{\partial\Psi_y}{\partial t}\right)\end{aligned} \tag{3.35}$$

3.4 Gradients for the stress-velocity code

The elastic forward problem and backpropagation of the data residuals is solved by using a time domain stress-velocity finite-difference (FD) code. Therefore the displacements in Eq. (3.35) have to be replaced by stresses and particle velocities $\mathbf{v}$ (Shipp and Singh [2002]):

$$\begin{aligned}
\delta\lambda' &= -\sum_{\text{sources}} \int \mathrm{dt} \left[\frac{(\sigma_{\mathrm{xx}} + \sigma_{\mathrm{yy}})(\Sigma_{\mathrm{xx}} + \Sigma_{\mathrm{yy}})}{4(\lambda+\mu)^2} \right] \\
\delta\mu' &= -\sum_{\text{sources}} \int \mathrm{dt} \left[\frac{\sigma_{\mathrm{xy}}\Sigma_{\mathrm{xy}}}{\mu^2} + \frac{1}{4}\left(\frac{(\sigma_{\mathrm{xx}} + \sigma_{\mathrm{yy}})(\Sigma_{\mathrm{xx}} + \Sigma_{\mathrm{yy}})}{(\lambda+\mu)^2} + \frac{(\sigma_{\mathrm{xx}} - \sigma_{\mathrm{yy}})(\Sigma_{\mathrm{xx}} - \Sigma_{\mathrm{yy}})}{\mu^2} \right) \right] \\
\delta\rho' &= -\sum_{\text{sources}} \int \mathrm{dt} \left[\frac{\partial \mathrm{v_x}}{\partial \mathrm{t}} \Psi_{\mathrm{x}} + \frac{\partial \mathrm{v_y}}{\partial \mathrm{t}} \Psi_{\mathrm{y}} \right]
\end{aligned} \tag{3.36}$$

where σ_{ij} and Σ_{ij} are the stresses of the forward and backpropagated wavefield, respectively. The displacements Ψ_{i} in the density gradient are calculated from the particle velocities by numerical integration.

3.5 Gradients for different model parametrizations

The gradients in terms of other material parameters $\mathrm{m_{new}}$ can be calculated by applying the chain rule on the Frechét kernel in the adjoint problem (Eq. (3.26)):

$$\delta\mathbf{m_{new}} = \sum_{\text{sources}} \int \mathrm{dt} \sum_{\mathrm{R}} \left[\frac{\partial \mathbf{u}}{\partial \mathbf{m}} \frac{\partial \mathbf{m}}{\partial \mathbf{m_{new}}} \right]^* \delta\mathbf{u} \tag{3.37}$$

Using the relationships between P-wave velocity $\mathrm{V_p}$, S-wave velocity $\mathrm{V_s}$, the Lamé parameters λ, μ and density ρ:

$$\mathrm{V_p} = \sqrt{\frac{\lambda + 2\mu}{\rho}},\ \mathrm{V_s} = \sqrt{\frac{\mu}{\rho}} \tag{3.38}$$

or

$$\lambda = \rho\mathrm{V_p^2} - 2\rho\mathrm{V_s^2},\ \mu = \rho\mathrm{V_s^2}. \tag{3.39}$$

The gradient for $\mathrm{V_p}$ can be written as:

$$\begin{aligned}
\delta\mathrm{V_p} &= \sum_{\text{sources}} \int \mathrm{dt} \sum_{\mathrm{R}} \left[\frac{\partial \mathbf{u}}{\partial \lambda} \frac{\partial \lambda}{\partial \mathbf{V_p}} + \frac{\partial \mathbf{u}}{\partial \mu} \frac{\partial \mu}{\partial \mathbf{V_p}} + \frac{\partial \mathbf{u}}{\partial \rho} \frac{\partial \rho}{\partial \mathbf{V_p}} \right]^* \delta\mathrm{u_i} \\
&= \sum_{\text{sources}} \int \mathrm{dt} \sum_{\mathrm{R}} \left[\frac{\partial \mathbf{u}}{\partial \lambda} 2\rho\mathrm{V_p} \right]^* \delta\mathrm{u_i} \\
&= 2\rho\mathrm{V_p} \sum_{\text{sources}} \int \mathrm{dt} \sum_{\mathrm{R}} \left[\frac{\partial \mathbf{u}}{\partial \lambda} \right]^* \delta\mathrm{u_i} \\
&= 2\rho\mathrm{V_p}\delta\lambda
\end{aligned} \tag{3.40}$$

The gradients for $\mathrm{V_s}$ and ρ are calculated in a similar way, so the gradients in terms of seismic velocities can be written as:

$$\begin{aligned}
\delta\mathrm{V_p} &= 2\rho\mathrm{V_p}\delta\lambda, \\
\delta\mathrm{V_s} &= -4\rho\mathrm{V_s}\delta\lambda + 2\rho\mathrm{V_s}\delta\mu, \\
\delta\rho_{\mathrm{vel}} &= (\mathrm{V_p^2} - 2\mathrm{V_s^2})\delta\lambda + \mathrm{V_s^2}\delta\mu + \delta\rho
\end{aligned} \tag{3.41}$$

3.6 Estimation of an optimum step length μ_{n}

The choice of the step length μ_{n} in Eq. 3.10 is crucial for the convergence of the steepest gradient optimization method. I demonstrate this using a very familiar test problem for optimization routines, the Rosenbrock function (Rosenbrock [1960], Fig. 3.4)

$$\mathrm{f_r(x,y)} = (1-\mathrm{x})^2 + 100(\mathrm{y}-\mathrm{x}^2)^2 \tag{3.42}$$

The aim is to find the minimum of this function loacted at the point [1,1] which is surrounded by a very narrow valley. We start the search for the minimum at [-0.5,0.5]. An obvious first choice would be a constant step length. Fig. 3.4 (top) shows the convergence after 16000 iteration steps of the steepest descent method when choosing a step length $\mu_{\mathrm{n}} = 2\mathrm{e}-3$. Note the large model update during the first iteration step, when the gradient of the Rosenbrock function is large. After reaching the narrow valley the gradient is much smaller and as a result the model updates are also decreasing. This leads to a very slow convergence speed. Especially near the minimum the model updates become very small. When choosing a larger step length ($\mu_{\mathrm{n}} = 2\mathrm{e}-3$, Fig. 3.4 (bottom)) the model update is larger even when the gradient is small, but the code fails to converge at all. Instead it is trapped in a narrow part of the valley. To solve this problem a variable step length is introduced. For three test step lengths μ_1, μ_2 and μ_3 three test models are calculated

$$\begin{aligned}
\mathbf{m_{test1}} &= \mathbf{m_n} + \mu_1 \delta \mathbf{m'_n} \\
\mathbf{m_{test2}} &= \mathbf{m_n} + \mu_2 \delta \mathbf{m'_n} \\
\mathbf{m_{test3}} &= \mathbf{m_n} + \mu_3 \delta \mathbf{m'_n}
\end{aligned} \tag{3.43}$$

and the corresponding L2-norms $\mathrm{L2_1}$, $\mathrm{L2_2}$ and $\mathrm{L2_3}$ are estimated (Fig. 3.5). The true misfit function (yellow line) can be approximated by fitting a parabola through the three points

$$\mathrm{L2_i} = \mathrm{a}\mu_{\mathrm{i}}^2 + \mathrm{b}\mu_{\mathrm{i}} + \mathrm{c} \tag{3.44}$$

where $\mathrm{i} \in \{1,2,3\}$ and a, b, c are the unkown coefficients. This system of equations can be written as matrix equation:

$$\begin{pmatrix} \mu_1^2 & \mu_1 & 1 \\ \mu_2^2 & \mu_2 & 1 \\ \mu_3^2 & \mu_3 & 1 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} L2_1 \\ L2_2 \\ L2_3 \end{pmatrix}$$

or

$$\mathbf{Ax} = \mathbf{b}. \tag{3.45}$$

The unknown coefficients of this matrix equation are formally defined by

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}, \tag{3.46}$$

In the FWT code the solution vector $\mathbf{x}$ is calculated by Gaussian elimination. In the following the step length at the extremum of the parabola is defined the extremum step length μ_{ext} (denoted as green square in Fig.3.5). This extremum step length is

$$\mu_{\mathrm{ext}} = -\frac{\mathrm{b}}{2\mathrm{a}}. \tag{3.47}$$

The application of the variable step length calculation to the Rosenbrock test problem is shown in Fig. 3.6. The number of required iteration steps to reach the minimum is reduced by a factor 4 when compared with the constant step length gradient method. The only problem remaining is the slow convergence speed in the small valley of the Rosenbrock function, due to the fact that the update occurs along the gradient direction of the objective function resulting in a "criss-cross" pattern. This behaviour can be avoided by applying a nonlinear conjugate gradient method (chapter 3.7). In case of the FWT algorithm the three test step lengths for the individual material parameters are calculated by scaling the maximum of the gradient to the maximum of the actual models:

$$\begin{aligned}
\mu_\lambda &= \mathrm{p}\frac{\max(\lambda_{\mathrm{n}})}{\max(\delta\lambda_{\mathrm{n}})} \\
\mu_\mu &= \mathrm{p}\frac{\max(\mu_{\mathrm{n}})}{\max(\delta\mu_{\mathrm{n}})} \\
\mu_\rho &= \mathrm{p}\frac{\max(\rho_{\mathrm{n}})}{\max(\delta\rho_{\mathrm{n}})}
\end{aligned} \tag{3.48}$$

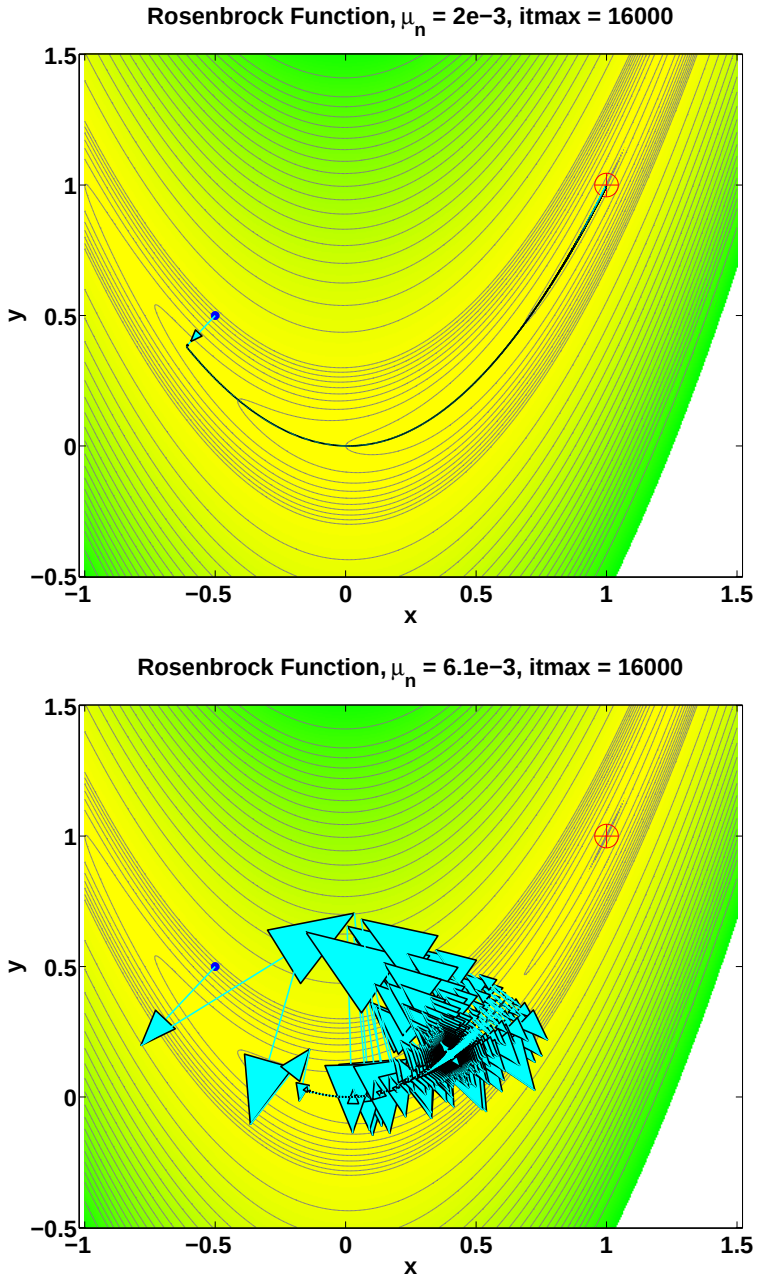


Figure 3.4: Results of the convergence test for the Rosenbrock function. The minimum is marked with a red cross, the starting point with a blue point. The maximum number of iterations is 16000. The step length μ_{n} varies between $2\mathrm{e}-3$ (top) and $6.1\mathrm{e}-3$ (bottom).

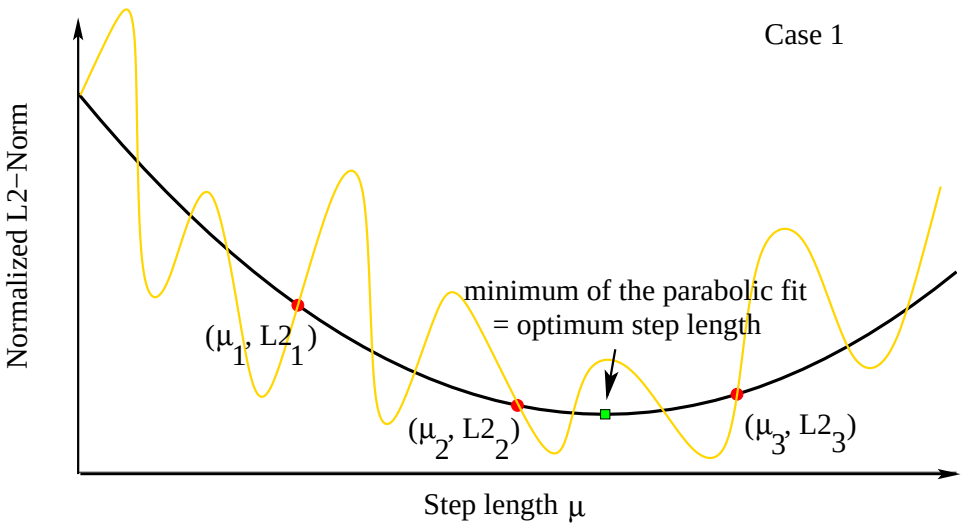


Figure 3.5: Line search algorithm to find the optimum step length $\mu_{\rm opt}$: The true misfit function (yellow line) is approximated by a parabola fitted by 3 points.

For most tests in the following chapters $\rm p_1 = 0.0025,\ p_2 = 0.005,\ p_3 = 0.01$, which corresponds to maximum model changes of 1/4, 1/2 and 1 %, worked very well for the optimum step length estimation. All material parameters are updated at the same time. To save computational time the corresponding $\rm L_2-$norms are calculated for a few representative shots (in most cases 3). For the acoustic case the step length estimation by parabolic fitting works very well and leads to a smooth decrease of the misfit function during the FWT (Kurzmann (2007), personal communication, Kurzmann et al. [2008]). For the multiparameter elastic FWT the misfit function consists of more local minima and therefore the decrease of the objective function is not as smooth as in the acoustic case. Brossier [2009] proposed a more intensive bracketing stage before applying the parabolic fit. For $\rm p_1 = 0.0$ the test step lengths $\rm p_2$ and $\rm p_3$ are calculated to satisfy the following criteria:

$$\begin{aligned} &\mathrm{L2_2}(\mathbf{m_{test2}} = \mathbf{m_n} + \mu_2 \delta \mathbf{m'_n}) < \mathrm{L2_1}(\mathbf{m_{test1}} = \mathbf{m_n}) \\ &\mathrm{L2_3}(\mathbf{m_{test3}} = \mathbf{m_n} + \mu_3 \delta \mathbf{m'_n}) > \mathrm{L2_2}(\mathbf{m_{test2}} = \mathbf{m_n} + \mu_2 \delta \mathbf{m'_n}) \end{aligned} \tag{3.49}$$

This approach leads to a smoother decrease of the objective function, but also increases the number of required forward models.

3.7 Nonlinear Conjugate Gradient Method

To increase the convergence speed in narrow valleys it would be better to update the model at iteration step n not exactly along the gradient direction $\delta \mathbf{m}_{\rm n}$, but along the conjugate direction $\delta \mathbf{c}_{\rm n}$

$$\delta \mathbf{c}_{\rm n} = \delta \mathbf{m}_{\rm n} + \beta_{\rm n} \delta \mathbf{c}_{\rm n-1}, \tag{3.50}$$

The first iteration step (n=1) consists of a model update along the steepest descent direction:

$$\mathbf{m}_2 = \mathbf{m}_1 + \mu_1 \delta \mathbf{m}_1, \tag{3.51}$$

For all subsequent iteration steps ($\rm n > 1$) the model is updated along the conjugate direction:

$$\mathbf{m}_{\rm n+1} = \mathbf{m}_{\rm n} + \mu_{\rm n} \delta \mathbf{c}_{\rm n}, \tag{3.52}$$

where $\delta \mathbf{c}_1 = \delta \mathbf{m}_1$. The weighting factor β can be calculated in different ways:

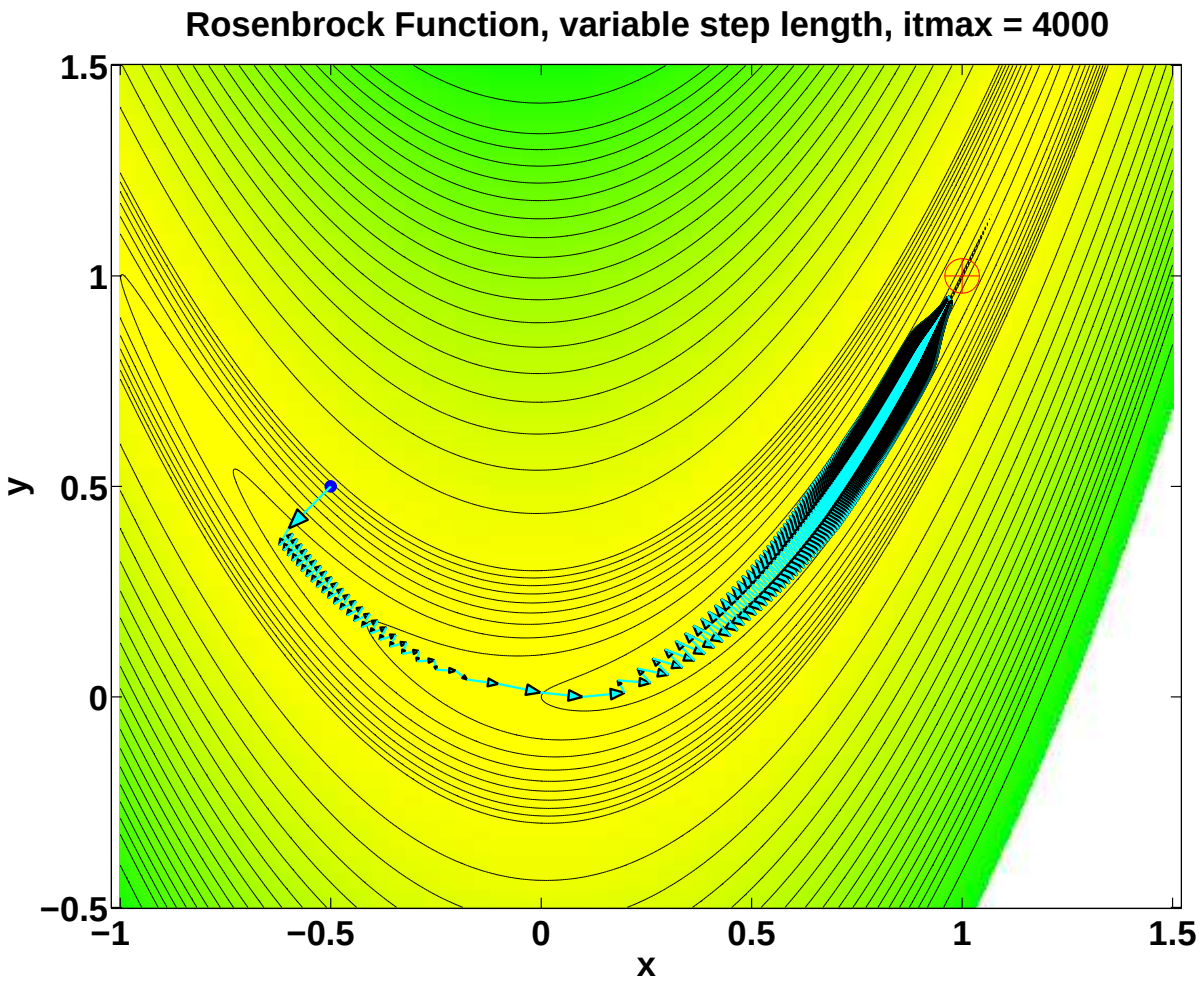


Figure 3.6: Results of the convergence test for the Rosenbrock function. The minimum is marked by a red cross, the starting point by a blue point. The maximum number of iterations is 4000. The optimum step length is calculated at each iteration by the parabola fitting algorithm. Note the criss-cross pattern of the updates in the narrow valley near the minimum.

1. Fletcher-Reeves:

$$\beta_{\rm n}^{\rm FR} = \frac{\delta\mathbf{m}_{\rm n}^{\rm T}\delta\mathbf{m}_{\rm n}}{\delta\mathbf{m}_{\rm n-1}^{\rm T}\delta\mathbf{m}_{\rm n-1}} \tag{3.53}$$

2. Polak-Ribiére:

$$\beta_{\rm n}^{\rm PR} = \frac{\delta\mathbf{m}_{\rm n}^{\rm T}(\delta\mathbf{m}_{\rm n} - \delta\mathbf{m}_{\rm n-1})}{\delta\mathbf{m}_{\rm n-1}^{\rm T}\delta\mathbf{m}_{\rm n-1}} \tag{3.54}$$

3. Hestenes-Stiefel:

$$\beta_{\rm n}^{\rm HS} = \frac{\delta\mathbf{m}_{\rm n}^{\rm T}(\delta\mathbf{m}_{\rm n} - \delta\mathbf{m}_{\rm n-1})}{\delta\mathbf{c}_{\rm n-1}^{\rm T}(\delta\mathbf{m}_{\rm n} - \delta\mathbf{m}_{\rm n-1})} \tag{3.55}$$

I use the very popular choice $\beta_{\rm n} = \max[0, \beta_{\rm n}^{\rm PR}]$ which provides an automatic direction reset. This is important because subsequent search directions lose conjugacy requiring the search direction to be reset to the steepest descent

direction. Note that the conjugate gradient method doesn't require any additional computational time because only the gradient $\delta \mathbf{m}_n$ at two subsequent iterations has to be known. The application of the nonlinear conjugate gradient method combined with the variable step length calculation to the Rosenbrock function is shown in Fig. 3.7. The criss-cross pattern of the steepest descent method has vanished. The conjugate gradient method converges already after 2000 iterations compared with 4000 iteration steps of the pure gradient method.

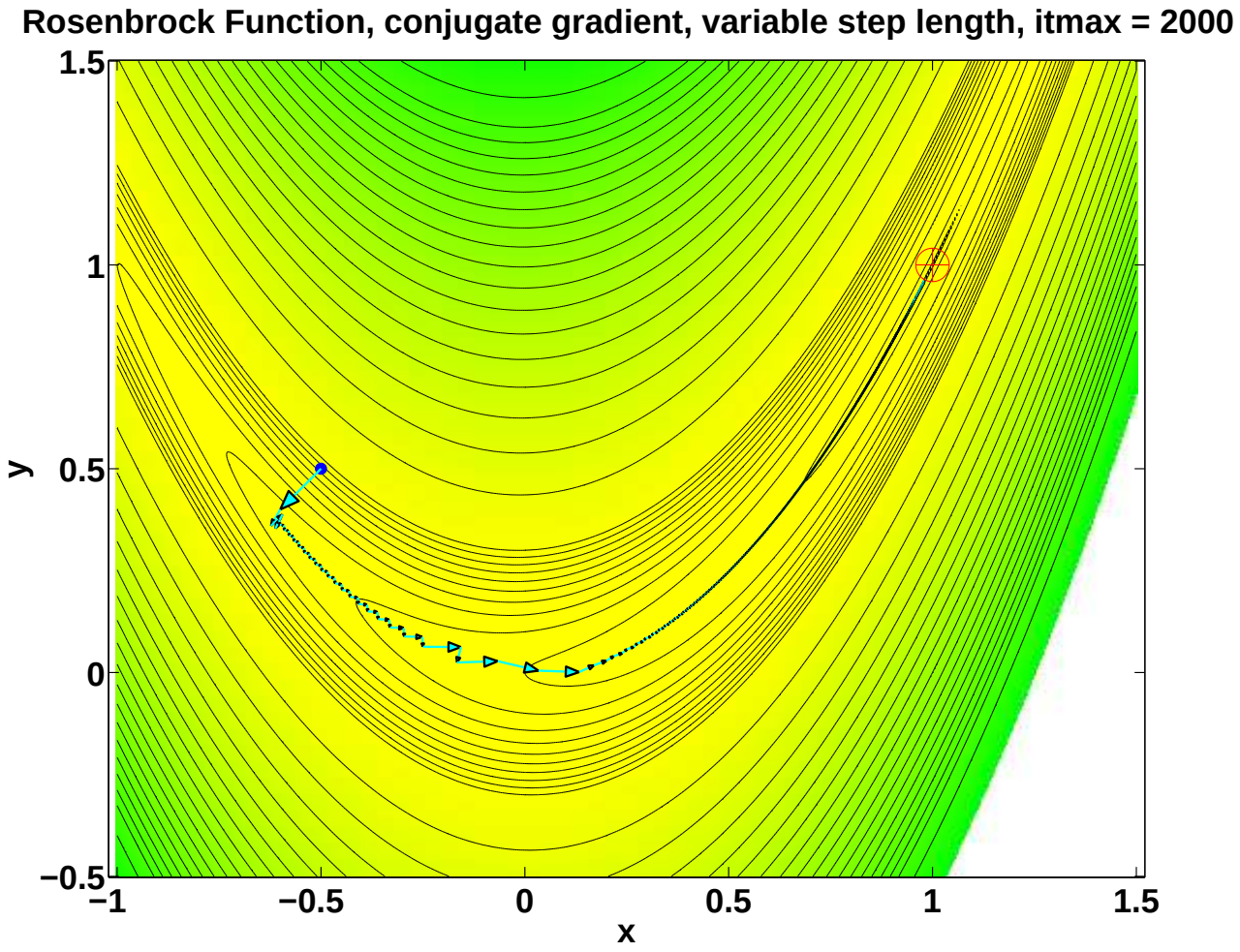


Figure 3.7: Results of the convergence test for the Rosenbrock function using the conjugate gradient method, where the optimum step length is calculated with the parabolic fitting algorithm. The minimum is marked by a red cross, the starting point by a blue point. The maximum number of iterations is 2000.

3.8 The elastic FWT algorithm

In summary the FWT algorithm consists of the following steps:

1. Define a starting model $\mathbf{m_1}$ in the parameter space. This model should represent the long wavelength part of the underground very well, because the FWT code is only capable to reconstruct structures at or below the dominant seismic wavelength due to its slow convergence speed, the nonlinearity of the problem and the inherent use of the Born approximation to calculate the gradient direction.

2. At iteration step n do:

   (a) For each shot solve the forward problem, stated in Eq.(3.16) for the actual model $\mathbf{m}_\mathrm{n}$ to generate a synthetic dataset $\mathbf{u^{mod}}$ and the wavefield $\mathbf{u}(\mathbf{x},\mathrm{t})$.

   (b) Calculate the residual seismograms $\delta\mathbf{u} = \mathbf{u^{mod}} - \mathbf{u^{obs}}$ for the x- and y-components of the seismic data.

   (c) Generate the wavefield $\mathbf{\Psi}(\mathbf{x},\mathrm{t})$ by backpropagating the residuals from the receiver postions.

   (d) Calculate the gradients $\delta\mathbf{m}_\mathrm{n}$ of each material parameter according to Eqs.(3.35).

   (e) To increase the convergence speed an appropriate preconditioning operator P is applied to the gradient $\delta\mathbf{m}$

   $$\delta\mathbf{m}_\mathrm{n}^\mathrm{p} = \mathrm{P}\delta\mathbf{m}_\mathrm{n} \tag{3.56}$$

   Examples of simple preconditioning operator are given in chapter 7.1.3 for a reflection geometry.

   (f) For a further increase of the convergence speed calculate the conjugate gradient direction for iteration steps $\mathrm{n} \geq 2$:

   $$\delta\mathbf{c}_\mathrm{n} = \delta\mathbf{m}_\mathrm{n}^\mathrm{p} + \beta\delta\mathbf{c}_{\mathrm{n}-1}, \text{ with } \delta\mathbf{c}_1 = \delta\mathbf{m}_1^\mathrm{p} \tag{3.57}$$

   where the weighting factor

   $$\beta^\mathrm{PR} = \delta\mathbf{m}_\mathrm{n}^\mathrm{p}\frac{\delta\mathbf{m}_\mathrm{n}^\mathrm{p} - \delta\mathbf{m}_{\mathrm{n}-1}^\mathrm{p}}{\delta\mathbf{m}_{\mathrm{n}-1}^\mathrm{p}\delta\mathbf{m}_{\mathrm{n}-1}^\mathrm{p}} \tag{3.58}$$

   by Polak-Ribiére is used. The convergence of the Polak-Ribiére method is guaranteed by choosing $\beta = \max[\beta^\mathrm{PR}, 0]$.

   (g) Estimate the step length μ_n by the line search algorithm described in chapter 3.6.

   (h) Update the material parameters using the gradient method

   $$\mathbf{m}_{\mathrm{n}+1} = \mathbf{m}_\mathrm{n} - \mu_\mathrm{n}\delta\mathbf{c}_\mathrm{n}. \tag{3.59}$$

   If the material parameters are not coupled by empirical relationships it is important to update all three elastic material parameters at the same time, otherwise strong artefacts may dominate the inversion result, especially in the case of very complex media.

3. If the residual energy E is smaller than a given value stop the iteration. Otherwise continue with the next iteration step.

Chapter 4

Source Time Wavelet Inversion

Introduction:
To remove the contribution of the unknown source time function (STF) from the waveform residuals, it is necessary to design a filter which minimizes the misfit to the field recordings and raw synthetics. The library libstfinv from Thomas Forbriger is exported from the Trac-/svn-System TFSoftware and can be used with a C API in DENISE. The purpose of this library is to provide methods for the derivation of source-time-functions in approaches to full waveform inversion. Given a set of recorded data and a set of synthetic data (typically, but not necessarilly the impulse response of the subsurface) a source time function is obtained due to some optimization citerion. The synthetic waveforms are convolved with this wavelet and the convolved synthetics as well as the wavelet itself are returned to the user.

The source time wavelet in this context not necessarily is the actual force time history of the source used in the experiment or a similar quantity of physical meaning. The source time wavelet simply is the wavelet which minimizes the misfit between synthetic and recorded waveforms due to some misfit condition, if the synthetics are concolved with this wavelet. In particular this implies that the synthetics not necessarily must be the impulse response (Greens function) of the subsurface, they may simply be synthetic waveform computed for some generic source wavelet (like a Ricker wavelet). The derived source time function then have to be understood with respect this generic wavelet.

The library provides different engines to find an optimal source time wavelet. The basic steps of operation are:

1. Initialize an engine. In this step pointers to arrays are passed to the engine together with some header information. The engines memorizes these pointers and expects to find the recorded data as well as the synthetics at the inidcated locations in memory.

2. Call the run()-function of the engine. The engine takes the recorded and synthetic data currently found at the memory arrays, calculates an optimzed wavelet and returns the wavelet together with the convolved synthetics by copying them to the memory locations inidicated by the initializer of the engine. This step is repeated after each computation of synthetic data.

3. Remove the engine once you terminate the iteration of inversion.

How to construct parameter strings:
A specific engine is selected by passing a parameter string to the library interface. This parameter string may further contain parameters to control the execution mode of the engine. The parameter string starts with an ID-sequence identifying the desired engine. In the parameter string the ID-sequence is terminated by a colon (:).

After selecting the desired engine, the interface function strips of the ID-sequence as well as the colon from the parameter string and initializes the engine, passing the references to user workspace as well as the rest of the parameter string. The rest of the parameter string may consist of several control parameters being separated by colons (:). Each control parameter may just be a flag (switch to turn an option on) or may come along with a parameter value. The value of the parameter is separated by an equal sign (=).

Examples:

- To select frequency domain least squares and shift the returned source time function by 0.4s and switch on verbose mode, pass the following parameter string:
  fdlsq:tshift=0.4:verbose
- To select frequency domain least squares, apply offset dependent weights and use a power of two to speed up the FFT:
  fdlsq:pow2:exp=1.4

Detailed description of the engine 'Fourier domain least squares (fdlsq)'
If

- d_{lk} is the Fourier coefficient of recorded data at Frequency f_l and receiver k at offset r_k,
- s_{lk} is the Fourier coefficient of the corresponding synthetics and
- q_l is that of the sought source tim function,

then this engine will minimize the objective function

$$E = \sum_{l,k} |w_{lk}\ (d_{lk} - s_{lk} q_l)|^2 + \sum_l \lambda^2 \, |q_l|^2 = \chi^2 + \psi^2 \tag{4.1}$$

with respect to the real part q_l' and the imaginary part q_l'' of

$$q_l = q_l' + i\, q_l''. \tag{4.2}$$

In the above expression

$$\chi^2 = \sum_{l,k} |w_{lk}\ (d_{lk} - s_{lk} q_l)|^2 \tag{4.3}$$

is the data misfit with weights w_{lk} and

$$\psi^2 = \sum_l \lambda^2 \, |q_l|^2 \tag{4.4}$$

is used for regularization and will introduce a water-level in the deconvolution. λ will balance both contributions. The conditions

$$\frac{\partial E}{\partial q_l'} \stackrel{!}{=} 0 \quad \wedge \quad \frac{\partial E}{\partial q_l''} \stackrel{!}{=} 0 \tag{4.5}$$

result in (Forbriger [2001], appendix A.3)

$$q_l = \frac{\eta^2 \sum\limits_k f_k^2 \, s_{kl}^* \, d_{kl}}{\lambda^2 + \eta^2 \sum\limits_k f_k^2 \, s_{kl}^* \, s_{kl}} \quad \forall\, l \tag{4.6}$$

where

$$w_{lk} = \eta \, f_k \tag{4.7}$$

and f_k is a receiver specific weighting factor. Now η and λ have to be used to balance the regularization. We aim to specify a waterlevel as a fraction of synthetic data energy.

Setting up the waterlevel:
The misfit equals one if the scaled energy of the residual $d_{lk} - s_{lk} q_l$ equals the scaled energy of the synthetics s_{lk} and

$$\eta^2 = \frac{1}{\sum\limits_k f_k^2 \sum\limits_l |s_{lk}|^2} \tag{4.8}$$

is the reciprocal of the scaled energy of the synthetics. If we then choose

$$\frac{\lambda^2}{\eta^2} = \frac{\epsilon^2}{N\eta^2} = \frac{\epsilon^2}{N} \sum_k f_k^2 \sum_{l=0}^{N-1} |s_{lk}|^2 \tag{4.9}$$

where N is the number of frequencies, then ϵ^2 will specify a waterlevel as a fraction of the scaled energy of the synthetics.

Using Parceval's Theorem to calculate signal energy:
Parceval's Theorem for a signal $a(t)$ and its Fourier transform $\tilde{a}(\omega)$ is

$$\int_{-\infty}^{+\infty} \left|a(t)\right|^2 \mathrm{d}t = \int_{-\infty}^{+\infty} \left|\tilde{a}(\omega)\right|^2 \frac{\mathrm{d}\omega}{2\pi}. \tag{4.10}$$

If S_{jk} are the time series samples corresponding to the Fourier coefficients $\tilde{s}_{lk}$ and Δt is the sampling interval then

$$\sum_{k=0}^{M-1} \left|S_{jk}\right|^2 \Delta t = \sum_{l=0}^{M-1} \left|\tilde{s}_{lk}\right|^2 \frac{1}{M\,\Delta t}, \tag{4.11}$$

where $M = 2N$ is the number of samples in the time series. In the above calculation the energy sum only uses the positive frequencies and

$$\sum_k f_k^2 \sum_{l=0}^{N-1} \left|\tilde{s}_{lk}\right|^2 = N\,(\Delta t)^2 \sum_k f_k^2 \sum_{j=0}^{2N-1} \left|S_{jk}\right|^2. \tag{4.12}$$

Fourier coefficients s_{lk} calculated by the stfinv::STFFourierDomainEngine are not scaled (see documentation of libfourierxx and libfftw3), such that

$$\Delta t\, s_{lk} = \tilde{s}_{lk} \tag{4.13}$$

(both, s_{lk} and $\tilde{s}_{lk}$ are Fourier coefficients). Consequently

$$\sum_k f_k^2 \sum_{l=0}^{N-1} \left|s_{lk}\right|^2 = N \sum_k f_k^2 \sum_{j=0}^{2N-1} \left|S_{jk}\right|^2. \tag{4.14}$$

Final calculation recipe:
The solution to our problem is

$$q_l = \frac{\sum_k f_k^2\, s_{lk}^*\, d_{lk}}{\epsilon^2 \sum_k f_k^2 \sum_{j=0}^{2N-1} \left|S_{jk}\right|^2 + \sum_k f_k^2\, s_{lk}^*\, s_{lk}} \quad \forall\, l, \tag{4.15}$$

where

$$\sum_{j=0}^{2N-1} \left|S_{jk}\right|^2 \tag{4.16}$$

is the sum of the squared sample values S_{jk} of the synthetic time series for receiver k, f_k are the scaling factors, and ϵ^2 is the water level parameter.
Author: Thomas Forbriger. See for more information the online doxygen documentation:
http://www.rz.uni-karlsruhe.de/ bi77/txt/cxx/libstfinv/html/index.html

Chapter 5

Getting Started

In the following sections, we give a short description of the different modeling parameters, options and how the program is used in a parallel MPI environment.

5.1 Requirements

The parallelization employs functions of the Message Passing Interface (MPI). MPI has to be installed when compiling and running the DENISE software. At least two implementations exist for Unix-based networks: OpenMPI and MPICH2. The LAM-MPI implementation is no longer supported by the developers. Currently all three implementation work with DENISE. OpenMPI and MPICH2 are MPI programming environments and development systems for heterogeneous computers on a network. With OpenMPI or MPICH2, a dedicated cluster or an existing network computing infrastructure can act as one parallel computer solving one problem. The latest version of OpenMPI can be obtained from http://www.open-mpi.org. MPICH2 is available at http://www-unix.mcs.anl.gov/mpi/mpich. LAM-MPI can be downloaded here: http://www.lam-mpi.org.

5.1.1 LAM

Even though outdated, LAM-MPI will be used to illustrate the setting up of the MPI implementation. In order to reproduce the following instructions, you must first install the LAM-MPI software. On SUSE LINUX systems the package can be installed with yast. The software can also be downloaded from http://www.lam-mpi.org. A good documentation of LAM/MPI is available at this site. Before you can run the DENISE software on more than one node you must boot LAM. This requires that you can connect to all of your nodes with ssh without a password. To enable ssh connection without a password you must generate authentication keys on each node using the command ssh-keygen -t rsa. These are saved in the file $\sim$/.ssh/id_rsa.pub. Copy all authentication keys into one file named authorized_keys and copy this file on all nodes into the directory $\sim$/.ssh.

Before you can boot LAM you must specify the IP addresses of the different processing elements in an ASCII file. An example file is par/lamhosts. Each line must contain one IP address. The first IP number corresponds to node 0, the second line to node 1 and so forth. Note that in older LAM-MPI implementations the mpirun command to run the FD programs must always be executed on node 0 !, i.e. you must log on node 0 (first line in the file par/lamhosts) and run the software on this node. In the last stable release of LAM-MPI, the node 0 just has to be listed in the lamhosts file (the order does not matter). To boot lam just do *lamboot -v par/lamhosts*. To run the software in serial on a single PC just type *lamboot* without specifying a lamhosts file. You can still specify more PEs in the FD software but all processes will be executed on the same CPU. Thus this only makes sense if you run the software on a multicore system. In this case, you should boot it without a lamhosts file and specify a total number of 2 processing elements (PEs). To shut down LAM again (before logout) use the command *lamclean -v*. However, it is not necessary to shut down/restart LAM after each logout/login.

5.1.2 How to run DENISE on the Linuxcluster at RZ Kiel

Before you can run DENISE on the Linux cluster at the computing centre in Kiel you have satisfy, that the different nodes can communicate. This has to be done only once.

1. Create a file in your home-directory mpd.conf, containing only one line:

   ```
   MPD_SECRETWORD=value
   ```

   For *value* enter an appropriate password. You can easily create this file by entering

   ```
   [sungwXXX@rzcluster ~]$ echo "MPD_SECRETWORD=value" > mpd.conf
   ```

   at the command line. It has to be satisfied, that only you have read and write access to this file, which can be achieved with the command:

   ```
   [sungwXXX@rzcluster ~]$ chmod 700 mpd.conf
   ```

2. Generate a pair of authentication keys for ssh with:

   ```
   [sungwXXX@rzcluster ~]$ ssh-keygen -t dsa
   ```

   You can accept the default values by hitting <return>.

3. Copy the file $HOME/.ssh/id_dsa.pub to $HOME/.ssh/authorized_keys.

4. Finally copy the file /usr/local/etc/known_hosts.aktuell to the directory $HOME/.ssh using the file name *known_hosts*.

Because DENISE can produce up to a few GB of data output, don't run the code from the home-directory. To submit a batch job it is required, that DENISE is located on one of the following drives: */work/username*, */work1/username*, */work2/username*, */work3/username* or */work4/username*. Keep in mind, that the file system on /workX will not be automatically backuped, so do a manual backup from time to time. To use different programming tools and software packages environment modules are used. Before you can load different modules you have to initialize it with

```
[sungwXXX@rzcluster ~]$ . /usr/local/Modules/3.2.6/init/bash
```

for the bash-shell. By typing

```
[sungwXXX@rzcluster ~]$ module avail
```

you get a list of all available packages. To compile and run the DENISE code you have to load a C-compiler and an MPI or MPICH implementation. You can load a module with

```
[sungwXXX@rzcluster ~]$ module load <module name>
```

I can recommend two possible configurations, which seem to work well together with DENISE. A PGI compiler together with MPICH2:

```
[sungwXXX@rzcluster ~]$ module load pgi11.4
[sungwXXX@rzcluster ~]$ module load mpich2-pgi
```

or a PGI compiler in combination with OpenMPI:

```
[sungwXXX@rzcluster ~]$ module load pgi11.4
[sungwXXX@rzcluster ~]$ module load openmpi-pgi
```

Stay away from the GNU-compilers, they have a very bad performance. You can unload any module by using

```
[sungwXXX@rzcluster ~]$ module unload <module name>
```

After compiling the code (see section 5.3), you can define and start a batch job with a shell script like this:

```
#!/bin/bash
#PBS -o DENISE.out
#PBS -j oe
#PBS -l walltime=24:00:00
#PBS -l select=2:marwin=true:ncpus=8:mpiprocs=8:mem=8gb
#PBS -q small
#PBS -N DENISE
#
. /usr/local/Modules/3.2.6/init/bash
module load mpich2-pgi
#
cd /work4/sungwXXX/DENISE/par
mpirun -np 16 -machinefile $PBS_NODEFILE /work4/sungwXXX/DENISE/bin/denise \
                                         /work4/sungwXXX/DENISE/par/DENISE.inp
qstat -f $PBS_JOBID
exit
```

The individual parameters and possible batch-job classes are described in more detail on the homepage of the RZ Kiel http://www.rz.uni-kiel.de/hpc/rzcluster/pbs/batchbetrieb.html. The most important parameters are

- *walltime*, which defines how long the job will actually run.
- *select* defines the number of requested nodes.
- *marwin=true* denotes that the exclusive nodes for the Institute of Geophysics should be used. If more than 160 cores are required you can also use *all=true*.
- *ncpus* the number of CPUs on each node.
- *mpiprocs* the number of MPI processes on each node.
- *mem* the required memory per node.
- *-q* the requested batch-class.

Note also that absolute path-descriptions have to be used within the batch-job file. An example for a job-file can be found in the /DENISE/jobs directory. The job can be submitted with

```
[sungwXXX@rzcluster ~]$ qsub DENISE.job
```

With

```
[sungwXXX@rzcluster ~]$ qstat
```

you can check the status of the Job. And with

```
[sungwXXX@rzcluster ~]$ qdel <job_id>
```

you can cancel a submitted or running job, where $< \mathrm{job_id} >$ denotes the number at the first column of the status information, f.e.

```
[sungwXXX@rzcluster ~]$ qstat
245229.rzcluster  DENISE   sungwXXX   00:00:00 R small
[sungwXXX@rzcluster ~]$ qdel 245229.rzcluster
```

For further information I refer to the home page of the RZ-Kiel: http://www.rz.uni-kiel.de/hpc/rzcluster/

5.2 Installation

After unpacking the software package (e.g. by *tar -zxvf DENISE.tgz*) and changing to the directory DENISE (*cd DENISE*) you will find different subdirectories:

bin
This directory contains all executable programs, generally DENISE and snapmerge. These executables are generated using the command *make <program>* (see below).

contrib
This directory contains external contributions to DENISE.

doc
This directory contains documentation on the software (this users guide) as well as some important papers in PDF format on which the software is based on (see above).

genmod
Contains the model and benchmark files for DENISE.

mfiles
Here some Matlab routines (m-files) are stored. These Matlab programs can be used to find optimal relaxation frequencies to approximate a constant Q (qapprox.m) or to plot Q as a function of frequency for certain relaxation frequencies and value of tau (qplot.m). For further details on the theory behind these algorithms we refer to the Phd thesis of T. Bohlen Bohlen [1998] and to the paper in which the so-called tau-method is described Blanch et al. [1995]. The parameter tau is used in this software to define the level of attenuation.

par
Parameter files for DENISE modeling.

scripts
Here, you will find examples of script-files used to submit modeling jobs on cluster-computers.

src
This directory contains the complete source codes. The following programs are available and my be compiled using make <program>.

5.3 Compilation of DENISE

Before compiling the main program DENISE you have to compile the required additional libraries e.g. for timedomain filtering, the inversion for the correction filter for the unknown source time function and so on. In the DENISE/par directory simply use:

```
-bash-2.05b$:~/DENISE/par> make
```

which should install the following libraries:

```
lib cseife
lib stfinv
lib aff
lib fourier
```

as well as the binary of DENISE itself. In contrib/Makefile_var there were several environment variables which are necessary to compile the libraries successfully. Furthermore, it is necessary to preinstall FFTW - Fastest Fourier Transform in the West (*http://www.fftw.org/*). Please check the successful installation in the folder contrib/header.

The source code of DENISE is located in the directory DENISE/src. To compile DENISE the name of the model function has to be entered in the src/MAKEFILE. Depending on your MPI environment (MPI distribution) you may need to modify the compiler options in src/Makefile. For a few typical platforms the compiler options are available in src/Makefile. It is often useful to enable a moderate level of optimization (typically -03). The highest level of optimization -O4 can lead to a strong performance improvement. For example the optimization option -04 of the hcc LAM compiler leads to a speedup of DENISE of approximately 30 per cent. Eventhough keep in mind that -O4 can also lead to crashes and compilation errors, when used in combination with certain compilers. No other changes in the Makefile should be necessary.

```
# Makefile for DENISE

#---------------------------------------------------------
# edit here:

# source code for model generation

MODEL_EL = half_space.c
EXEC= ../bin


# Compiler (LAM: CC=hcc, CRAY T3E: CC=cc)

# ON Linux cluster running LAM (options for DENISE)
#CC=hcc
#LFLAGS=-lm -lmpi -lcseife
#CFLAGS=-O3
#SFLAGS=-L./../libcseife
#IFLAGS=-I./../libcseife

# On CRAY T3E
# CC=cc

# On MARWIN
CC=mpicc
LFLAGS=-lm -lcseife
CFLAGS=-O3
SFLAGS=-L./../libcseife
IFLAGS=-I./../libcseife

# On HLRN system
#CC=mpcc
#LFLAGS=-lm

# ALTIX
#CC=icc
#CFLAGS=-mp -O3 -ipo
#LFLAGS=-lmpi -lm -i-static

# On the workstations in Karlsruhe (GPI)
CC=mpicc
LFLAGS=-lm -lcseife -lstfinv -laff -lfourierxx -lfftw3 -lstdc++
CFLAGS=-O3
SFLAGS=-L./../libcseife -L./../contrib/bin
IFLAGS=-I./../libcseife -I./../contrib/header

# after this line, no further editing should be necessary
# ---------------------------------------------------------
```

The program snapmerge that is used to merge the snapshots (see below) can be compiled with ”make snapmerge” in the directory /src. Since this is not a MPI program (no MPI functions are called) the MPI libraries are not required and any standard compiler (like gcc and cc) can be used to compile this program. The executables denise and snapmerge are located in the directory /bin.

5.4 Running the program

Each DENISE run reads the required parameters from a parameter file par/DENISE.json. A detailed description of the parameters are described in chapter 6. The command to start a simulation on 8 processor with the lowest priority of -19 (in order to allow working on the a workstation while running a simulation) is as follows. Please note, that we assume you have navigated to the folder DENISE/par.

```
mpirun -np 8 nice -19 ../bin/denise DENISE.json
```

If you use LAMMPI the command *lamboot -v lamhost* must be executed on node 0 which is the PE where ./par/lamhosts is the file containing IP addresses of all computing nodes. It is often useful to save the standard output of the program for later reference. The screen output may be saved to DENISE.out using

```
mpirun -np 8 nice -19 ../bin/denise DENISE.json > DENISE.out
```

After the output of geometry and model parameters the code starts the time stepping and displaying timing information:

```
================================================================================

 MYID=0 *****  Starting simulation (forward model) for shot 1 of 1  **********

================================================================================

 Number of samples (nts) in source file: 3462
 Number of samples (nts) in source file: 3462
 Message from function wavelet written by PE 0
 1 source positions located in subdomain of PE 0
 have been assigned with a source signal.

 PE 0 outputs source time function in SU format to start/source_signal.0.su.shot1

 Computing timestep 1000 of 3462

 **Message from update_v (printed by PE 0):
 Updating particle velocities ... finished (real time: 0.00 s).
 particle velocity exchange between PEs ... finished (real time: 0.00 s).

 **Message from update_s (printed by PE 0):
 Updating stress components ... finished (real time: 0.00 s).
 stress exchange between PEs ... finished (real time: 0.00 s).
 total real time for timestep 1000 : 0.01 s.

 Computing timestep 2000 of 3462

 **Message from update_v (printed by PE 0):
 Updating particle velocities ... finished (real time: 0.00 s).
 particle velocity exchange between PEs ... finished (real time: 0.00 s).

 **Message from update_s (printed by PE 0):
 Updating stress components ... finished (real time: 0.00 s).
 stress exchange between PEs ... finished (real time: 0.00 s).
 total real time for timestep 2000 : 0.01 s.

 Computing timestep 3000 of 3462

 **Message from update_v (printed by PE 0):
 Updating particle velocities ... finished (real time: 0.00 s).
 particle velocity exchange between PEs ... finished (real time: 0.00 s).

 **Message from update_s (printed by PE 0):
 Updating stress components ... finished (real time: 0.00 s).
 stress exchange between PEs ... finished (real time: 0.00 s).
 total real time for timestep 3000 : 0.01 s.
 PE 0 is writing 11 seismograms (vx) to
         su/DENISE_US_x.su.shot1.it1
 PE 0 is writing 11 seismograms (vy) to
         su/DENISE_US_y.su.shot1.it1

 **Info from main (written by PE 0):
```

```
CPU time of program per PE: 17 seconds.
Total real time of program: 18.08 seconds.
Average times for
velocity update:        0.003 seconds
stress update:          0.002 seconds
velocity exchange:      0.000 seconds
stress exchange:        0.000 seconds
timestep:       0.005 seconds
```

5.5 Postprocessing

The wavefield snapshots can be merged using the program *snapmerge*. The program snapmerge is not a MPI program. Therefore, it can be executed without MPI and the mpirun command. You can run snapmerge on any PC since a MPI environment (e.g. LAM) is not required. You may therefore copy the snapshot outputs of the different nodes to another non-MPI computer to merge the files together. *snapmerge* reads the required information from the DENISE parameter file. Simply type

```
-bash-2.05b$~/DENISE/par> ../bin/snapmerge DENISE.json
```

Depending on the model size the merge process may take a few seconds or hours. For the simple block model it only takes a few seconds. The output should read like this:

```
pressure (files: ./snap/test.bin.p.??? ).

writing merged snapshot file to  ./snap/test.bin.p
Opening snapshot files: ./snap/test.bin.p.???  ... finished.
Copying... ... finished.
Use
xmovie n1=100 n2=100 < ./snap/test.bin.p loop=1 label1=Y label2=X title=%g
to play movie.
```

Chapter 6

Parameter definition with json input file

The geometry of the FD grid and all parameters for the wave field simulation and inversion have to be defined in a parameter file. DENISE uses a parameter file according to the JSON standard wich is described in this chapter. In the following we will first list a full input file for a forward modeling as an example and later explain every input parameter section in detail:

```
#-------------------------------------------------------------
#      JSON PARAMETER FILE FOR DENISE
#-------------------------------------------------------------
# description:
# description/name of the model: model grid created by ../genmod/2layer.c
#

{
"Domain Decomposition" : "comment",
"NPROCX" : "4",
"NPROCY" : "2",

"FD order" : "comment",
"FDORDER" : "2",
"MAXRELERROR" : "0",

"2-D Grid" : "comment",
"NX" : "500",
"NY" : "100",
"DH" : "0.2",

"Time Stepping" : "comment",
"TIME" : "0.5",
"DT" : "5.0e-05",

"Source" : "comment",
"QUELLART" : "4",
"SIGNAL_FILE" : "./ormsby.dat",

"QUELLTYP" : "3",
"ANGLE" : "0.0",

"SOURCE_FILE" : "./source/sources.dat",
"RUN_MULTIPLE_SHOTS" : "1",
```

```
"Model" : "comment",
"READMOD" : "0",
"MFILE" : "model/model_Test",

"Free Surface" : "comment",
"FREE_SURF" : "1",

"PML Boundary" : "comment",
"FW" : "20",
"DAMPING" : "600.0",
"FPML" : "31.25",
"BOUNDARY" : "0",
"npower" : "4.0",
"k_max_PML" : "8.0",

"Receiver" : "comment",
"SEISMO" : "1",
"READREC" : "1",
"REC_FILE" : "./receiver/receiver.dat",
"REFRECX, REFRECY" : "0.0 , 0.0",
"XREC1, YREC1" : "6.0 , 0.2",
"XREC2, YREC2" : "93.0 , 0.2",
"NGEOPH" : "80",

"Seismograms" : "comment",
"SEIS_FORMAT" : "1",
"SEIS_FILE_VX" : "su/DENISE_x.su",
"SEIS_FILE_VY" : "su/DENISE_y.su",


"General inversion parameters" : "comment",
"INVMAT1" : "1",
"INVMAT" : "10",
}
```

All lines in the parameter file are formated according to the JSON standard (www.json.org) and organized as follows:

```
"VARNAME" = "Parameter value",
```

where VARNAME denotes the name of the global variable in which the value is saved in all functions of the program. A comment line can look like this:

```
"Comment" = "This is a useful comment",
"2D Grid information" = "comment",
```

Sometimes possible values are described in comments, feel free to add own comments. Basically all non JSON conform line will be ignored. The order of parameters can be arbitrarily organized. The built-in JSON parser will search for the need parameters and displays found values. If critical parameters are missing the code will stop and an error message will appear.

If you use the json input file some default values for the forward modeling and the inversion are set. The default values are written in the following subsections in red. The input file `DENISE_FW_all_parameters.json` in the directory `par/in_and_out` is an input file for a forward modeling containing all parameters that can be defined. Analog to that the input file `DENISE_INV_all_parameters.json` is an example for an inversion input file. The

input files `DENISE_FW.json` and `DENISE_INV.json` contain only the parameters that must be set by the user.

In the beginning of the code development DENISE used a different kind of input parameter file. The current version is still able to read this old input file. The old parameter file was organized as follows:

```
description_of_parameter_(VARNAME)_(switches) = parameter value
```

where VARNAME denotes the name of the global variable in which the value is saved in all functions of the program. The possible values are described in switches. A comment line is indicated by a # on the very first position of a line. You can find an example of such a parameter file in `par/in_and_out/DENISE_HESSIAN.inp`. You can switch between the two possible input files via the file extension. Use ".inp" as file extension to read in the old input file or the file extension ".json" to use the new input file.

6.1 Domain decomposition

```
"Domain Decomposition" : "comment",
"NPROCX" : "4",
"NPROCY" : "2",
```

Parallelization is based on domain decomposition (see Figure 6.1), i.e each processing element (PE) updates the wavefield within his portion of the grid. The model is decomposed by the program into sub grids. After decomposition each processing elements (PE) saves only his sub-volume of the grid. NPROCX and NPROCY specify the number of processors in x-, y-direction, respectively (Figure 6.1). The total number of processors thus is NP=NPROCX*NPROCY. This value must be specified when starting the program with the mpirun command: *mpirun -np <NP> ../bin/DENISE DENISE.json* (see section 5.4). If the total number of processors in DENISE.json and the command line differ, the program will terminate immediately with a corresponding error message. Obviously, the total number of PEs (NPROCX*NPROCY) used to decompose the model should be less equal than the total number of CPUs which are available on your parallel machine. If you use LAM and decompose your model in more domains than CPUs are available two or more domains will be updated on the same CPU (the program will not terminate and will produce the correct results). However, this is only efficient if more than one processor is available on each node. In order to reduce the amount of data that needs to be exchanged between PEs, you should decompose the model into more or less cubic sub grids. In our example, we use 2 PEs in each direction: NPROCX=NPROCY=2. The total number of PEs used by the program is NPROC=NPROCX*NPROCY=4.

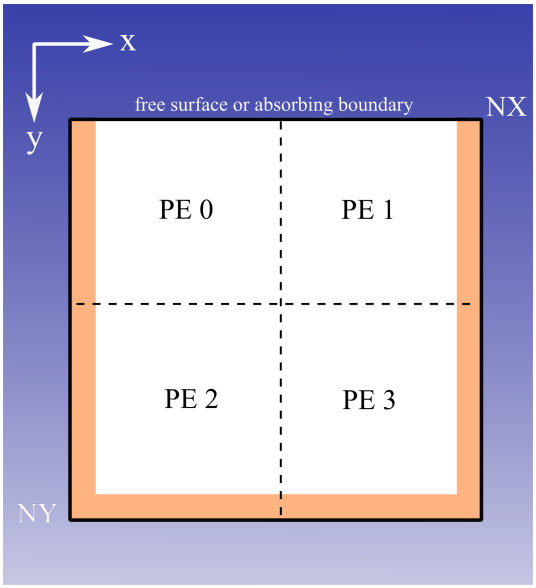


Figure 6.1: Geometry of the numerical FD grid using 2 processors in x-direction (NPROCX=2) and 2 processors in y-direction (NPROCY=2). Each processing element (PE) is updating the wavefield in its domain. At the top of the numerical mesh the PEs apply a free surface boundary condition if FREE_SURF=1, otherwise an absorbing boundary condition (PML). The width of the absorbing frame is FW grid points. The size of the total grid is NX grid points in x-direction and NY gridpoints in y-direction. The size of each sub-grid thus is NX/NPROCX x NY/NPROCY gridpoints. The origin of the Cartesian coordinate system (x,y) is at the top left corner of the grid.

6.2 Discretization

```
"2-D Grid" : "comment",
"NX" : "500",
"NY" : "100",
"DH" : "0.2",
```

These lines specify the size of the total numerical grid (Figure 6.1). NX and NY give the number of grid points in the x- and y-direction, respectively, and DH specify the grid spacing in x- and y-direction. The size of the total internal grid in meters in x-direction is NX*DH and in y-direction NY*DH. To allow for a consistent domain decomposition NX/NPROCX and NY/NPROCY must be integer values.

To avoid numerical dispersion the wavefield must be discretized with a certain number of gridpoints per wavelength. The number of gridpoints per wavelength required, depends on the order of the spatial FD operators used in the simulation (see section 2.3.1). In the current FD software, 2nd, 4th, 6th and 8th order operators are implemented. The criterion to avoid numerical dispersion reads:

$$DH \leq \frac{v_{s,min}}{2 f_c n} \tag{6.1}$$

where $\frac{v_{s,min}}{2f_c}$ is the smallest wavelength propagating through the model. $v_{s,min}$ denotes the minimum shear wave velocity in the model, and $f_c = 1/TS$ is the center frequency of the source wavelet. The program assumes that the maximum frequency of the source signal is approximately two times the center frequency. The center frequency is approximately one over the duration time TS. The value of n for different FD operators is tabulated in table 2.2. The criterion 6.1 is checked by the FD software. If the criterion is violated a warning message will be displayed in the DENISE output section "— CHECK FOR GRID DISPERSION —". Please note, that the FD-code will NOT terminate due to grid dispersion, only a warning is given in the output file.

6.3 Order of the FD operator

```
"FD order" : "comment",
"FDORDER" : "2",
"MAXRELERROR" : "0",
```

The order of the used FD operator is defined by the option FDORDER (FDORDER=2, 4 6 or 8). With the option MAXRELERROR the user can switch between Taylor (MAXRELERROR=0) and Holberg (MAXRELERROR=1-4) FD coefficients of different accuracy. The chosen FD operator and FD coefficients have an influence on the numerical stability and grid dispersion (see chapter 2.3.1).

6.4 Time stepping

```
"Time Stepping" : "comment",
"TIME" : "0.5",
"DT" : "5.0e-05",
```

The propagation time of seismic waves in the entire model is TIME (given in seconds). The time stepping interval (DT in s) has to fulfill the stability criterion (2.22) in section 2.3.2. The program checks these criteria for the entire model, outputs a warning message if these are violated , stops the program and will output the time step interval for a stable model run.

6.5 Sources

```
"Source" : "comment",
"QUELLART" : "4",
"QUELLART values: ricker=1;fumue=2;from_SIGNAL_FILE=3;SIN**3=4;Gaussian_deriv=5;Spike=6;fr
"SIGNAL_FILE" : "./ormsby.dat",

"QUELLTYP" : "3",
"QUELLTYP values (point_source): explosive=1;force_in_x=2;force_in_y=3;rotated_force=4" :
"ANGLE" : "0.0",

"SRCREC" : "1",
"SRCREC values : read source positions from SOURCE_FILE=1, PLANE_WAVE=2" : "comment",

"SOURCE_FILE" : "./source/sources.dat",
"RUN_MULTIPLE_SHOTS" : "1",

"PLANE_WAVE_DEPTH" : "0.0",
"PHI" : "0.0",
"TS" : "0.032",


Default values are:
SRCREC=1
```

5 built-in wavelets of the seismic source are available. The corresponding time functions are defined in src/wavelet.c. You may modify the time functions in this file and recompile to include your own analytical wavelet or to modify the shape of the built-in wavelets.

Ricker wavelet (QUELLART=1):

$$r(\tau) = \left(1 - 2\tau^2\right)\exp(-\tau^2) \quad \text{with} \quad \tau = \frac{\pi(t - 1.5/f_c - t_d)}{1.0/f_c} \tag{6.2}$$

Fuchs-Müller wavelet (QUELLART=2):

$$f_m(t) = \sin(2\pi(t-t_d)f_c) - 0.5\sin(4\pi(t-t_d)f_c) \quad \text{if} \quad t \in [t_d, t_d + 1/fc] \quad \text{else} \quad fm(t) = 0 \tag{6.3}$$

sin^3 wavelet (QUELLART=4):

$$s3(t) = 0.75\pi f_c \sin(\pi(t+t_d)f_c)^3 \quad \text{if} \quad t \in [t_d, t_d + 1/fc] \quad \text{else} \quad s3(t) = 0 \tag{6.4}$$

First derivative of a Gaussian function (QUELLART=5):

$$f(t) = -2.0a(t-t_s)\exp(-a(t-t_s)^2) \quad \text{with} \quad a = \pi^2 f_c^2 \quad \text{and} \quad t_s = 1.2/f_c \tag{6.5}$$

Delta pulse (QUELLART=6): Lowpass filtered delta pulse. Note, that it is not clear if the lowpass filter used in the current version works correctly for a delta pulse.

In these equations, t denotes time and $f_c = 1/TS$ is the center frequency. t_d is a time delay which can be defined for each source position in SOURCE_FILE. Note that the symmetric (zero phase) Ricker signal is always delayed by $1.0/f_c$, which means that after one period the maximum amplitude is excited at the source location. Three of these 5 source wavelets and the corresponding amplitude spectra for a center frequency of $f_c = 50$ Hz and a delay of $t_d = 0$ are plotted in Figure 6.2. Note the delay of the Ricker signal described above. The Fuchs-Müller wavelet has a slightly higher center frequency and covers a broader frequency range.

You may also use your own time function as the source wavelet (for instance the signal of the first arrival recorded by a geophone at near offsets). Specify QUELLART=3 and save the samples of your source wavelet in ASCII-format in SIGNAL_FILE. SIGNAL_FILE should contain one sample per line. It should thus look like:

```
0.0
0.01
0.03
...
```

The time interval between the samples must equal the time step interval (DT) of the FD simulation (see above)! Therefore it might be necessary to resample/interpolate a given source time function with a smaller sample rate. You may use the matlab script mfiles/resamp.m to resample your external source signal to the required sampling interval.

It is also possible to read different external source wavelets for each shot. Specify QUELLART=7 and save the wavelets in su format in SIGNAL_FILE.shot<no_of_shot>. The wavelets in each su file must equal the time step intervel (DT) and the number of time steps of the FD simulation!

The following source types are availabe: explosive sources that excite compressional waves only (QUELLTYP=1), and point forces in the x- and y-direction (QUELLTYP=2,3). The force sources excite both P- and S-waves. The explosive source is located at the same position as the diagonal elements of the stress tensor, i.e. at (i,j) (Figure 2.1). The forces are located at the same position as the corresponding components of particle velocity (Figure 2.1). If (x,y) denotes the position at which the source location is defined in source.dat, then the actual force in x-direction is located at (x+DX/2,y) and the actual force in y-direction is located at (x,y+DY/2). With QUELLTYP=4 a custom directive force can be defined by a force angle between y and x. The angel of the force must be specified in the SOURCE_FILE after AMP. This force is not aligned along the main directions. The parameter ANGLE is without any use.

The locations of multiple sources must be defined in an external ASCII file (SOURCE_FILE) that has the following format:

```
NSRC
  %  XSRC ZSRC YSRC TD FC AMP   SOURCE_AZIMUTH SOURCE_TYPE (NSRC lines)
```

In the following lines, you can define certain parameters for each source point:
The first line must be the overall number of sources (NSRC). XSRC is the x-coordinate of a source point (in meter), YSRC is the y-coordinate of a source point (in meter). ZSRC is the z-coordinate should always be set to 0.0, because DENISE is a 2D code. TD is the excitation time (time-delay) for the source point [in seconds], FC is the center frequency of the source signal [in Hz], and AMP is the maximum amplitude of the source signal.

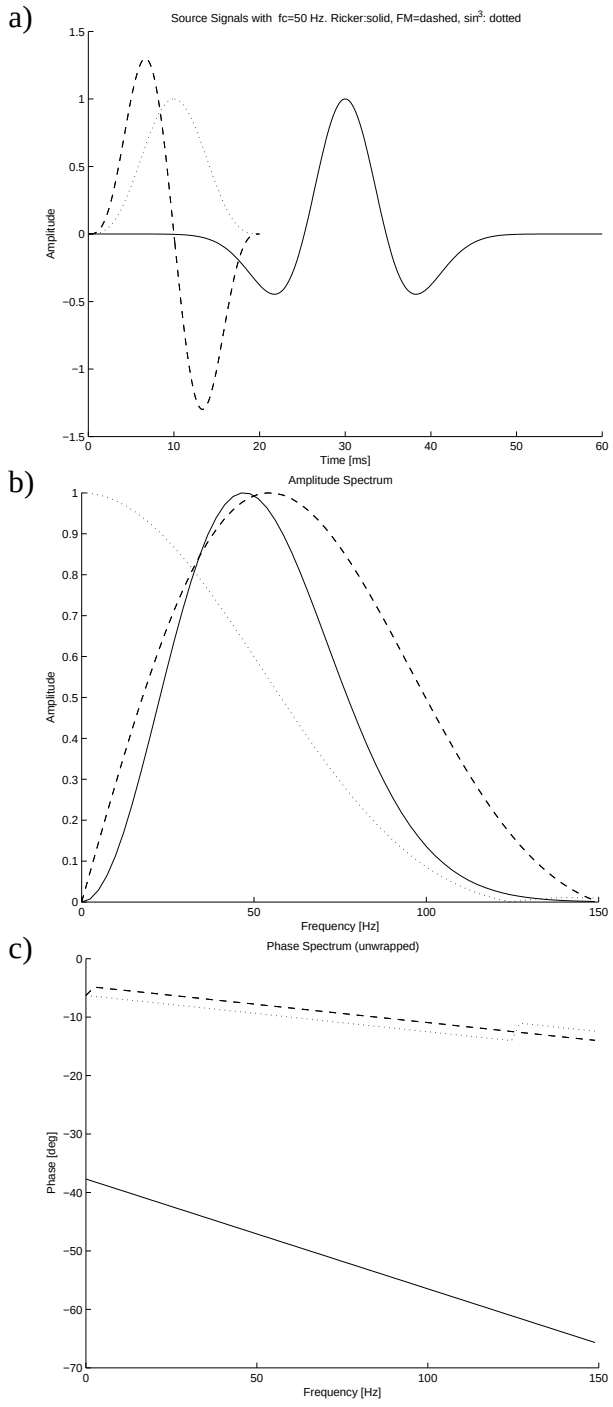


Figure 6.2: Plot of built-in source wavelets (equations 6.2, 6.3, 6.4) for a center frequency of $f_c = 50$ Hz ($TS = 1/f_c = 0.02$s): Ricker signal (solid), Fuchs-Müller signal (dashed), sin^3-signal (dotted). a) Time function, b) amplitude spectrum, c) phase spectrum.

Optional parameter: The SOURCE_AZIMUTH if SOURCE_TYPE is 4. The SOURCE_AZIMUTH is the angle between the y- and x-direction and with SOURCE_TYPE if SOURCE_TYPE is set here, the value of SOURCE_TYPE in the input file is ignored.

The SOURCE_FILE = ./sources/source.dat that defines an explosive source at $x_s = 2592.0$ m and $y_s = 2106.0$ m with a center frequency of 5 Hz (no time delay) is

```
2592.0        0.0        2106.0        0.0           5.0           1.0
```

If the option RUN_MULTIPLE_SHOTS=0 in the parameter file all shot points defined in the SOURCE_FILE are excitated simultaneously in one simulation. Setting RUN_MULTIPLE_SHOTS=1 will start individual model runs from i=1 to i=NSRC with source locations and properties defined at line i of the SOURCE_FILE. (To apply a full waveform inversion you have to use RUN_MULTIPLE_SHOTS=1.)

Instead of a single source or multiple sources specified in the SOURCE_FILE, you can also specify to excite a plane wave parallel (or tilted by an angle PHI) to the top of the model. This plane wave is approximated by a plane of single sources at every grid point at a depth of PLANE_WAVE_DEPTH below. The center source frequency f_c is specified by the inverse of the duration of the source signal TS. QUELLART and QUELLTYP are taken from the parameters as described above. If you choose the plane wave option by specifying a PLANE_WAVE_DEPTH>0, the parameters SRCREC and SOURCE_FILE will be ignored.

This option will not be supported in future releases of DENISE.

6.6 Model input

```
"Model" : "comment",
"READMOD" : "0",
"MFILE" : "model/test",
```

If READMOD=1, the P-wave, S-wave, and density model grids are read from external binary files. MFILE defines the basic file name that is expanded by the following extensions: P-wave model: ".vp", S-wave model: ".vs", density model: ".rho". In the example above, the model files thus are: "model/test.vp" (P-wave velocity model),"model/test.vs" (S-wave velocity model), and "model/test.rho" (density model).

In these files, each material parameter value must be saved as 32 bit (4 byte) native float. Velocities must be in meter/second, density values in $\mathrm{kg/m^3}$. The fast dimension is the y direction. See src/readmod.c. The number of samples for the entire model in the x-direction is NX, the number of values in the y-direction is NY. The file size of each model file thus must be NX*NY*4 bytes. You may check the model structure using the SU command ximage: ximage n1=<NY> < model/test.vp .

It is also possible to read Qp, and Qs grid files to allow for spatial variable attenuation. For this you must uncomment a few lines in src/readmod.c and generate the corresponding binary files.

If READMOD=0 the model is generated "on the fly" by DENISE, i.e. it is generated internally before the time loop starts. See `genmod/1D_linear_gradient_el.c` for an example function that generates a simple model with a linear vertical gradient "on the fly". If READMOD=0 this function is called in `denise.c` and therefore must be specified in src/Makefile (at the top of src/Makefile, see section 5.3). If you change this file, for example to change the model structure, you need to re-compile DENISE by changing to the src directory and "make denise".

6.7 Free surface

```
"Free Surface" : "comment",
"FREE_SURF" : "1",
```

A plane stress free surface is applied at the top of the global grid if FREE_SURF!=0 using the imaging method proposed by Levander [1988]. Note that the free surface is always located at y=0 or at the first grid point, respectively.

6.8 Boundary conditions

```
"PML Boundary" : "comment",
"FW" : "20",
"DAMPING" : "600.0",
"FPML" : "31.25",
"BOUNDARY" : "0",
"npower" : "4.0",
"k_max_PML" : "8.0",
```

The boundary conditions are applied on each side face and the bottom face of the model grid. If FREE_SURF = 0 the boundary conditions are also applied at the top face of the model grid. Note that the absorbing frames are always located INSIDE the model space, i.e. parts of the model structure are covered by the absorbing frame, in which no physically meaningful wavefield propagates. You should therefore consider the frame width when you design the model structure and the acquisition geometry (shot and receivers should certainly be placed outside).

A convolutional perfectly matched layer (CPML) boundary condition is used. The PML implementation is based on the following papers Komatitsch and Martin [2007] and Martin and Komatitsch [2009]. A width of the absorbing frame of FW=10-20 grid points should be sufficient. For the optimal realization of the PML boundary condition you have to specify the dominant signal frequency FPML occurring during the wave simulation. This is usually the center source frequency FC specified in the source file. DAMPING specifies the attenuation velocity in m/s within the PML. DAMPING should be approximately the propagation velocity of the dominant wave near the model boundaries.

In some cases, it is usefull to apply periodic boundary conditions (see section 2.2.3). IF BOUNDARY=1 no absorbing boundaries are installed at the left/right sides of the grid. Instead, wavefield information is copied from left to right and vice versa. The effect is, for example, that a wave which leaves the model at the left side enters the model again at the right side.

6.9 Receivers

```
"Receiver" : "comment",
"SEISMO" : "1",
"READREC" : "1",
"REC_FILE" : "./receiver/receiver.dat",
"REFRECX, REFRECY" : "0.0 , 0.0",
"XREC1, YREC1" : "6.0 , 0.2",
"XREC2, YREC2" : "93.0 , 0.2",
"NGEOPH" : "80",
```

If SEISMO>0, seismograms are saved on hard disk. If SEISMO equals 1 x- and y-component of particle velocity will be written according to parameters specified in Chapter 6.10. If SEISMO==2 pressure (sum of the diagonal components of the stress tensor) recorded at the receiver locations (receivers are hydrophones!) is written. if SEISMO=3 the curl and divergence are saved.

The curl and divergence of the particle velocities are useful to separate between P- and S-waves in the snapshots of the wavefield. DENISE calculates the divergence and the magnitude of the curl of the particle velocity field according to Dougherty and Stephen [1988]. The motivation for this is as follows. According to Morse and Feshbach Morse and Feshbach [1953] the energy of P- and S-wave particle velocities is, respectively,

$$E_p = (\lambda + 2\mu)\,(div(\vec{v}))^2 \quad \text{and} \quad E_s = \mu\,|rot(\vec{v})|^2 \quad . \tag{6.6}$$

λ and μ are the Lamè parameters, and $\vec{v}$ is the particle velocity vector.

The locations of the receivers may either be specified in a separate file REC_FILE or in this parameter file. If READREC=1 receiver locations are read from the ASCII-file REC_FILE. Each line contains the coordinates of one receiver, the first two number in each line indicate the horizontal x- and the vertical y-coordinate of each receiver

position. To give an example of a receiver file, the following 3 lines specify 3 receivers located at constant depth (2106.0 m). However, the receiver coordinates change in x-direction (starting at 540 m) and therefore lining up along the x-axis.

```
540.0   2106.0
1080.0  2106.0
1620.0  2106.0
```

These receiver coordinates in REC_FILE are shifted by REFREC[1], REFREC[2] into the x- and y-direction, respectively. This allows for completely moving the receiver spread without modifying REC_FILE. This may be useful for the simulation of moving profiles in reflection seismics.

If READREC=0 the receiver locations must be specified in the parameter file. In this case, it is assumed that the receivers are located along a straight line. The first receiver position is defined by (XREC1, YREC1), and the last receiver position by (XREC1, YREC1). The spacing between receivers is NGEOPH grid points.

Receivers are always located on full grid indices, i.e. a receiver that is located between two grid points will be shifted by the FD program to the closest next grid point. It is not yet possible to output seismograms for arbitrary receiver locations since this would require a certain wavefield interpolation.

It is important to note that the actual receiver positions defined in REC_FILE or in DENISE.json may vary by DX/2 and/or DY/2 and/or DZ/2 due to the staggered positions of the particle velocities and stress tensor components.

In our example, we specify 100 receiver location. Due to the small size of the model, most of the specified receiver positions will be located inside this absorbing boundary (if ABS=2, see Chapter 6.8). A corresponding warning message will appear. If you choose to read the receiver location from REC_FILE *./receiver/receiver.dat* (READREC=1), only 10 receivers locations are considered. The list of receivers specified in file *./receiver/receiver.dat* is equivalent to the parameters in the input file with only a larger distance between adjacent receivers (NGEOPH = 10.)

6.10 Seismograms

```
"Seismograms" : "comment",
"NDT" : "1",
"SEIS_FORMAT" : "1",
"SEIS_FILE_VX" : "su/DENISE_x.su",
"SEIS_FILE_VY" : "su/DENISE_y.su",
"SEIS_FILE_CURL" : "su/DENISE_rot.su",
"SEIS_FILE_DIV" : "su/DENISE_div.su",
"SEIS_FILE_P" : "su/DENISE_p.su",


Default values are:
NDT=1
```

If SEISMO>0 seismograms recorded at the receiver positions are written to the corresponding output files. The sampling rate of the seismograms is NDT*DT seconds. In case of a small time step interval and a high number of time steps, it might be useful to choose a high NDT in order to avoid a unnecessary detailed sampling of the seismograms and consequently large files of seismogram data. Keep in mind that the application of FWT requires NDT=1. Possible output formats of the seismograms are SU, ASCII and BINARY. It is recommended to use SU format for saving the seismograms. The main advantage of this format is that the time step interval (NDT*DT) and the acquisition geometry (shot and receiver locations) are stored in the corresponding SU header words. Also additional header words like offset are set by DENISE. This format thus facilitates a further visualization and processing of the synthetic seismograms. Note, however, that SU cannot handle sampling rates smaller than 1.0e-6 seconds and the number of samples is limited to about 32.000. In such cases, you should increase the sampling rate by increasing NDT. If this is impossible (for example because the Nyquist criterion is violated) you must choose a different output format (ASCII or binary).

6.11 Q-approximation

```
"Q-approximation",
"L" : "0",
"FL1" : "50.0",
"FL2" : "100.0",
"TAU" : "0.00001",
```

```
Default values are:
L=0
```

These lines may be used to define an overall level of intrinsic (viscoelastic) attenuation of seismic waves. In case of L=0, a purely elastic simulation is performed (no absorption). The frequency dependence of the (intrinsic) Quality factor $Q(\omega)$ is defined by the L relaxation frequencies (FL=$f_l = 2\pi/\tau_{\sigma l}$) and one value τ (see equation 5 in Bohlen [2002]). For a single relaxation mechanism (L=1) $Q \approx 2/\tau$ [Bohlen, 1998, Blanch et al., 1995, Bohlen, 2002]. If the model is generated "on the fly" the value of TAU can be assigned to all gridpoints for both P- and S-waves. Thus, intrinsic attenuation is homogeneous and equal for P- and S-waves ($Q_p(\omega) = Q_s(\omega)$). However, it is possible to simulate any spatial distribution of absorption by assigning the gridpoints with different Q-values by reading external grid files for Q_p (P-waves) and Q_s (S-waves) (see src/readmod.c) or by generating these files "on the fly" (see section 6.6 or exemplary model function `genmod/1D_linear_gradient_visc.c`).

Small Q values ($Q < 50$) may lead to significant amplitude decay and velocity dispersion. Please note, that due to dispersive media properties the viscoelastic velocity model is defined for the reference frequency only. In denise, this reference frequency is specified as the center source frequency. At the exact reference frequency, elastic and viscoelastic models are equivalent. As a consequence, slightly smaller and larger minimum and maximum velocity values occure in the viscoelastic model.

The frequency dependence of attenuation, i.e. Q and phase velocity as a function of frequency, may be calculated using the Matlab functions in the directory mfiles.

6.12 Wavefield snapshots

```
"Snapshots" : "comment",
"SNAP" : "0",
"TSNAP1" : "2.7e-3",
"TSNAP2" : "6.0",
"TSNAPINC" : "0.12",
"IDX" : "1",
"IDY" : "1",
"SNAP_FORMAT" : "3",
"SNAP_FILE" : "./snap/waveform_forward",
```

```
Default values are:
SNAP=0
IDX=1
IDY=1
```

If SNAP>0, wavefield information (particle velocities (SNAP=1=, pressure (SNAP=2), or curl and divergence of particle velocities (SNAP=3), or everything (SNAP=4)) for the entire model is saved on the hard disk (assure that enough free space is on disk!). Each PE is writing his sub-volume to disk. The filenames have the basic filename SNAP_FILE plus an extension that indicates the PE number in the logical processor array, i.e. the PE with number PEno writes his wavefield to SNAPFILE.PEno. The first snapshot is written at TSNAP1 seconds of seismic wave traveltime to the output files, the second at TSNAP1+TSNAPINC seconds etc. The last snapshots contains wavefield at TSNAP2 seconds. Note that the file sizes increase during the simulation. The snapshot files might become quite

LARGE. It may therefore be necessary to reduce the amount of snapshot data by increasing IDX, IDY and/or TSNAP-INC. In order to merge the separate snapshot of each PE after the comletion of the wave modeling, you can use the program snapmerge (see Chapter 5.2, section **src**). The bash command line to merge the snapshot files can look like this: *../bin/snapmerge DENISE.json*.

6.13 Receiver array

```
"Receiver array" : "comment",
"REC_ARRAY" : "0",
"REC_ARRAY_DEPTH" : "70.0",
"REC_ARRAY_DIST" : "40.0",
"DRX" : "4",

Default values are:
REC_ARRAY=0
```

These options are obsolete. Receiver arrays have to be defined directly in receiver.dat.

6.14 Monitoring the simulation

```
"Monitoring the simulation" : "comment",
"LOG_FILE" : "log/2layer.log",
"LOG" : "1",

Default values are:
LOG=1
LOG_FILE="log/LOG_FILE"
```

DENISE can output a lot of useful information about the modeling parameters and the status of the modeling process etc. The major part of this information is output by PE 0. If LOG=1, PE 0 writes this info to stdout, i.e. on the screen of your shell. This is generally recommended to monitor the modeling process. You may want to save this screen info to an output file by adding ”> DENISE.out” or ”ltee DENISE.out”. to your starting command. If LOG=1 all other processes with PE number (PEno) greater than zero will write their information to LOG_FILE.PEno. If you specify LOG=2 PE 0 will also output information to LOG_FILE.0. As a consequence only little information is written directly to the screen of your shell. On supercomputers where you submit modeling jobs to a queuing system as batch jobs LOG=2 may be advantageous. In case of LOG=2, you may still watch the simulation by checking the content of LOG_FILE.0 for example by using the Unix commands more or tail. After finishing the program the timing information is written to the ASCII file log/test.log.0.timings. This feature is useful to benchmark your local PC cluster or supercomputer. If LOG=0 no output from node 0 will be written, neither to stdout nor to an LOG file. There will be also no output of timing information to the ASCII file log/test.log.0.timings.

6.15 Checkpointing

```
"Checkpoints" : "comment",
"CHECKPTREAD" : "0",
"CHECKPTWRITE" : "0",
"CHECKPTFILE" : "tmp/checkpoint_fdveps",

Default values are:
CHECKPTREAD=0
CHECKPTWRITE=0
CHECKPTFILE="tmp/checkpoint_fdveps"
```

These options are obsolete and will not be supported in the current version of DENISE.

6.16 General inversion parameters

```
"General inversion parameters" : "comment",
"ITERMAX" : "10",
"DATA_DIR" : "su/measured_data/DENISE_real",
"INVMAT1" : "1",
"INVMAT" : "0",
"INVTYPE" : "2",
"QUELLTYPB" : "1",
"MISFIT_LOG_FILE" : "L2_LOG.dat",
"VELOCITY" : "0",

"Inversion for density" : "comment",
"INV_RHO_ITER" : "0",

"Inversion for Vp" : "comment",
"INV_VP_ITER" : "0",

"Inversion for Vs" : "comment",
"INV_VS_ITER" : "0",

"Cosine taper" : "comment",
"TAPER" : "0",
"TAPERLENGTH" : "10",

Default values are:
INVTYPE=2
MISFIT_LOG_FILE=L2_LOG.dat
VELOCITY=0
INV_RHO_ITER=0
INV_VP_ITER=0
INV_VS_ITER=0
TAPER=0
TAPERLENGTH=10
```

This section covers some general inversion parameters. The maximum number of iterations are defined by ITERMAX. The switch INVMAT controls if only the forward modeling code should be used (INVMAT=10), e. g. to calculate synthetic seismograms or a complete FWT run (INVMAT=0). The seismic sections of the real data are located in the DATA_DIR. As noted in section 3.5 the gradients can be expressed for different model parameterizations. The switch INVMAT1 defines which parameterization should be used, seismic velocities and density (Vp,Vs,rho, INVMAT1=1), seismic impedances (Zp,Zs,rho, INVMAT1=2) or Lamé parameters (λ, μ, ρ, INVMAT1=3). If models are read from binary files appropriate file extensions are required for the different models (see section 6.6). Depending on the data different components of the seismic sections can be backpropagated. For two component data (x- and y-component) set QUELLTYPB=1, only the y-component (QUELLTYPB=2) and only the x-component (QUELLTYPB=3).
During the inversion the misfit values are saved in a log file. The variable MISFIT_LOG_FILE sets the file name of this file. The log file consists of eight or nine columns and each line corresponds to one iteration step. The used step length is written in the first column. In the second to fourth column the three test step lengths used for the step length estimation are saved. The corresponding misfit values for these test step lengthes and the test shots are written to column five to seven. Column eight corresponds to the total misfit for all shots and if you use freqeuncy filtering then the ninth column corresponds to the corner frequency of the lowpass filter used in the inversion step.
In general DENISE tries to minimize the misfit in the particle displacement between the observed data and the synthetic data. If you set the switch VELOCITY to 1 the misfit in the particle velocity seismograms is minimized.
The parameters INV_RHO_ITER, INV_VP_ITER and INV_VS_ITER define from which inversion step on an inversion for density, Vp and Vs, respectively, is applied. To invert for density, Vp and Vs from the beginning of an

inversion set INV_RHO_ITER, INV_VP_ITER and INV_VS_ITER, respectively, to 0 or 1.

The parameters TAPER, TAPERLENGTH and INVTYPE are debug parameters and should not be changed.

6.17 Output of inversion results

```
"Output of inverted models" : "comment",
"INV_MODELFILE" : "model/model_Test",
"nfstart" : "1",
"nf" : "1",

"Output of gradients" : "comment",
"JACOBIAN" : "jacobian/jacobian_Test",
"nfstart_jac" : "1",
"nf_jac" : "1",
```

The inverted models are saved in INV_MODELFILE. The first model that is saved is at iteration step nfstart and then every $\mathrm{nf}^{\mathrm{th}}$ iteration step. Analog the gradients are saved every $\mathrm{nf_jac}^{\mathrm{th}}$ iteration step from iteration step nftart_jac on in JACOBIAN.

6.18 Change optimization method

```
"Hessian and Gradient-Method" : "comment",
"HESSIAN" : "0",
"FC_HESSIAN" : "100",
"ORDER_HESSIAN" : "4",
"TSHIFT_back" : "0.0",
"GRAD_METHOD" : "1",


Default values are:
HESSIAN=0
```

DENISE contains the option to calculate the diagonal elements of the Hessian (after Sheen et al. [2006]) for the starting model. Currently the Hessian is calculated for the Vp- and Vs-model and only for sources with directed forces in y-direction. The Hessian for the density model is not implemented yet.

The estimation of the Hessian requires for each shot position the solution of the forward problem with the actual source wavelet and for each receiver position the backpropagation of a spike. The forward and backpropagated wavefields have to be saved in memory and convolved with each other for each shot-receiver combination. This convolution is currently implemented as a multiplication in the frequency domain in conv_fd.c. Because the time intensive computation of the forward/backpropgated wavefields and the convolution process it is highly recommended to estimate the Hessian only for acquisition geometries with a small number of sources and receivers.

To calculate the Hessian the parameter HESSIAN in the input file has to be set to 1 and INVMAT to 0. If HESSIAN is set to 0 and INVMAT to 0 the code runs a standard FWT. If HESSIAN is set to 0 and INVMAT to 10 the code runs a forward modeling only.

A few important things to keep in mind, when calculating the Hessian in the current version of DENISE:

1. The HESSIAN can currently only be calculated for sources with forces in y-direction. Therefore the parameters QUELLTYP and QUELLTYPB in the input file should be set to 3 and 2 respectively.

2. To calculate the impulse response of a spike from the receiver positions a spike wavelet is defined as option in the parameter QUELLART. Due to the limitation of the frequency range by the grid dispersion criterion it

is not possible to calculate a true spike response, neither in time or in frequency-domain FD. Therefore, the spike wavelet is low-pass filtered using the parameters FC_HESSIAN and ORDER_HESSIAN. Note that in the current version there maybe is a bug in the definition of the spike wavelet. Therefore, one should check the source signal that will be written out using output_source_signal.c. The parameter TSHIFT_back must be given in seconds and causes a timeshift of the delta impulses which are propagated from the receiver positions into the medium. This parameter must be used in cases where the source signal of the forward propagated wavefields does not have its main impulse at t=0 s.

3. The estimated Vp-Hessian is written in binary format to JACOBIAN_hessian.old in the Jacobian directory, the estimated Vs-Hessian to JACOBIAN_hessian_u.old, respectively.

4. The inversion of the Hessian and the application of a Marquardt-Levenberg regularization can be done with the Matlab script check_hessian.m, which can be found in the /mfiles directory. With this script the value of the Marquardt factor, the damping function of the Hessian within the PML regions and additional preconditioning operators can be applied on the Hessian. To test the influence of the different parameters on the Hessian it is useful to calculate one FWT iteration and multiply the resulting gradient in check_hessian.m with the inverse Hessian. When the regularization and preconditioning of the inverse Hessian is satisfying, the inverse Hessian is written in binary format to the file taper.bin.

5. To apply the taper/inverse Hessian defined in taper.bin, on the gradient in DENISE, the parameter SWS_TAPER_FILE has to be set to 1 (see section 6.23). Each model parameter requires a taper file which should be located in the /par directory and named as taper.bin for the Vp-model, taper_u.bin for the Vs-model and taper_rho.bin for the density model. Because the density-Hessian is not implemented yet, taper_rho.bin can be simply a copy of taper.bin.

6. At each preconditioned conjugate gradient (PCG) iteration the Hessian for the starting model is multiplied by the regularized and preconditioned inverse Hessian. This scaling of the gradient improves the convergence speed of DENISE by approximately a factor 2. However it is not a real Gauss-Newton method, which would require the calculation of the inverse Hessian at each iteration step. The implementation of a quasi-Newton method, the L-BFGS method (see f.e. the book Nocedal and Wright [1999]) is currently in development. A pre-alpha version is included in this version. It can be activated by the parameter GRAD_METHOD, but a bug seem to prevent the convergence of the L-BFGS method. Therefore it is higly recommended to use only the PCG method (GRAD_METHOD=1) and not the L-BFGS method.

6.19 Misfit definition

```
"Gradient calculation" : "comment",
"LNORM" : "2",
"NORMALIZE" : "0",
"DTINV" : "2",
```

With LNORM=2 the L2 norm is used as misfit definition. In this case the misfit is scaled with the energy of the observed seismograms.
With LNORM=5 the global correlation is used as misfit function. It was suggested e. g. by Choi and Alkhalifah [2012] and consists of a zero-lag cross correlation of two normalized signals. The misfit is calculated by

$$E = -\sum_{i}^{ns}\sum_{j}^{nr}\sum_{k}^{nc}\frac{\vec{u}_{i,j,k}\cdot\vec{d}_{i,j,k}}{|\vec{u}_{i,j,k}||\vec{d}_{i,j,k}|} \tag{6.7}$$

where the sum over i denotes the sum over the sources, the sum over j denotes the sum over the receivers and the sum over k denotes the sum over the components used in the inversion. $\vec{u}$ are the forward modeled data and $\vec{d}$ are the observed data. The misfit is minimized but the misfit function is not yet normalized. Therefore, a perfect fit results in a misfit of $(-1)\cdot ns\cdot nr\cdot nc$.

LNORM=1 uses the L1 norm, LNORM=3 the Cauchy norm and LNORM=4 the SECH norm.
LNORM=7 uses the normalized L2 norm

$$E = \frac{\sum_i^{ns}\sum_j^{nr}\sum_k^{nc}\left|\frac{\vec{u}_{i,j,k}}{|\vec{u}_{i,j,k}|} - \frac{\vec{d}_{i,j,k}}{|\vec{d}_{i,j,k}|}\right|^2}{\sum_i^{ns}\sum_j^{nr}\sum_k^{nc}\left|\frac{\vec{d}_{i,j,k}}{|\vec{d}_{i,j,k}|}\right|^2} \tag{6.8}$$

suggested by Choi and Alkhalifah [2012]. The misfit is scaled with the energy of the observed seismograms.

If NORMALIZE is set to 1, the synthetic data and the measured data will be normalized before calculating the residuals.

To reduce the memory requirements during an inversion one can define that only every DTINV time sample is used for the calculation of the gradients. To set this parameter appropriately one has to keep in mind the Nyquist criterion to avoid aliasing effects.

6.20 Step length estimation

```
"Step length estimation" : "comment",
"EPS_SCALE" : "0.01",
"STEPMAX" : "4",
"SCALEFAC" : "4.0",
"TESTSHOT_START , TESTSHOT_END , TESTSHOT_INCR" : "1 , 2 , 1",
```

For the step length estimation a parabolic line search method proposed by Sourbier et al. [2009a,b], Brossier [2009] and Nocedal and Wright [1999] is implemented. For this step length estimation only two further test forward modelings are needed. The vector L2t contains the misfit values and the vector epst contains the corresponding step length. During the forward modeling of an iteration step the misfit norm of the data residuals is calculated for the shots defined by TESTSHOT_START, TESTSHOT_END and TESTSHOT_INC. The value L2t(1) then contains the misfit from the forward modeling and the corresponding epst(1) value is 0.0.

The step lengths for the different parameters are defined as:
EPSILON = EPS_SCALE * m_max/grad_max EPSILON = epst[i] * m_max/grad_max
where m_max is the maximum value of the corresponding model parameter in the whole model and grad_max is the maximum absolute value of the gradient.

For a better definition of the parabola the improved line search is now trying to estimate a steplength epst(2) with L2t(2)<L2t(1). If the code is not able to find an appropiate steplength using the user-defined value EPS_SCALE (f.e. EPS_SCALE = 0.01 = 1% change in terms of m_max/grad_max), the code divides this steplength by the variable SCALEFAC and calculates the misfit norm again. If this search fails after STEPMAX attempts DENISE exits with an error message. If the algorithm has found an appropriate value for epst(2), it is trying to estimate a steplength epst(3) with L2t(3)> L2t(2), by increasing the steplength

EPS_SCALE += EPS_SCALE/SCALEFAC.

If a corresponding value epst(3) can be found after STEPMAX forward modellings, DENISE can fit a parabola through the 3 points (L2t(i),epst(i)) and estimates an optimum step length at the minimum of the parabola. If the L2-value L2t(3) after STEPMAX forward models is still smaller than L2t(2) the optimum steplength estimated by parabolic fitting will be not larger than epst(3).

6.21 Abort criterion

```
"Termination of the programmme" : "comment",
"PRO" : "0.01",
```

Additionally to the parameter ITERMAX a second abort criterion is implemented in DENISE which is using the relative misfit change within the last two iterations. The relative misfit of the current iteration step and the misfit of the second to last iteration step is calculated with regard to the misfit of the second to last iteration step. If this relative change is smaller than PRO the inversion aborts or in case of using frequency filtering (TIME_FILT==1) it increases the corner frequency of the low pass filter and therefore switches to next higher bandwidth.

6.22 Minimum number of iteration per frequency

```
"Minimum number of iteration per frequency" : "comment",
"MIN_ITER" : "10",


Default values are:
MIN_ITER=0
```

If you are using frequeny filtering (TIME_FILT==1) during the inversion, you can set a minimum number of iterations per frequency. Within this minimum number of iteration per frequency the abort criterion PRO will receive no consideration.

6.23 Definition of the gradient taper geometry

```
"Definition of gradient taper geometry" : "comment",
"SWS_TAPER_GRAD_VERT" : "0",
"SWS_TAPER_GRAD_HOR" : "0",
"GRADT1 , GRADT2 , GRADT3 , GRADT4" : "5 , 15 , 490 , 500",
"SWS_TAPER_GRAD_SOURCES" : "0",
"SWS_TAPER_CIRCULAR_PER_SHOT" : "0",
"SRTSHAPE" : "1",
"SRTRADIUS" : "5.0",
"FILTSIZE" : "1",
"SWS_TAPER_FILE" : "0",
"SWS_TAPER_FILE_PER_SHOT" : "0",
"TAPER_FILE_NAME" : "taper.bin",
"TAPER_FILE_NAME_U" : "taper_u.bin",
"TAPER_FILE_NAME_RHO" : "taper_rho.bin",


Default values are:
SWS_TAPER_GRAD_VERT=0
SWS_TAPER_GRAD_HOR=0
SWS_TAPER_GRAD_SOURCES=0
SWS_TAPER_CIRCULAR_PER_SHOT=0
SWS_TAPER_FILE=0
SWS_TAPER_FILE_PER_SHOT=0
```

Different preconditioning matrices can be created and applied to the gradients (using the function 'taper_grad.c'). To apply a vertical or a horizontal taper one has to set the switches SWS_TAPER_GRAD_VERT and SWS_TAPER_GRAD_HOR to 1, respectively. The parameters for the vertical and the horizontal window are defined by the input file paramters GRADT1, GRADT2, GRADT3 and GRADT4. Please have a look at the function taper_grad.c directly to obtain more information about the actual definition of the tapers. It is also possible to apply cylindrical tapers around the source positions. This can be done by setting the switch SWS_TAPER_GRAD_SOURCES or SWS_TAPER_CIRCULAR_PER_SHOT to 1. If one uses SWS_TAPER_GRAD_SOURCES=1 only the final gradients (that means the gradients obtained by the summation of the gradients of each shots) are multiplied with a taper that decreases the gradients at all shot positions. Therefore, one looses the update information at the source positions. To avoid this one can use

SWS_TAPER_CIRCULAR_PER_SHOT=1. In this case the gradients of the single shots are preconditioned with a window that only decreases at the current shot position. This is done before the summation of all gradients to keep model update information also at the shot positions. The actual tapers are generated by the function 'taper_grad.c' and 'taper_grad_shot.c, respectively. The circular taper around the source positions decrease from a value of one at the edge of the taper to a value of zero at the source position. The shape of the decrease can be defined by an error function (SRTSHAPE=1) or a log-function (SRTSHAPE=2). The radius of the taper is defined in meter by SRTRADIUS. Note, that this radius must be at least 5 gridpoints. With the parameter FILTSIZE one can extend the region where the taper is zero around the source. The taper is set to zero in a square region of (2*FILTSIZE+1 times 2*FILTSIZE+1) gridpoints. All preconditioning matrices that are applied are saved in the par directory with the file names taper_coeff_vert.bin, taper_coeff_horz.bin and taper_coeff_sources.bin.

To apply an externally defined taper on the gradients in DENISE, the parameter SWS_TAPER_FILE has to be set to 1. Each model parameter requires a taper file which should be located in the /par directory and named as taper.bin for the Vp-model, taper_u.bin for the Vs-model and taper_rho.bin for the density model.

It is also possible to apply externally defined taper files to the gradients of the single shots before summation of these gradients. This can be done by setting SWS_TAPER_FILE_PER_SHOT to one. DENISE expects the tapers in TAPER_FILE_NAME.shot<shotnumber> for the lambda or vp gradients, in TAPER_FILE_NAME_U.shot<shotnumber> for the mu or vs gradients and in TAPER_FILE_NAME_RHO.shot<shotnumber> for the density gradients.

6.24 Definition of spatial filtering of the gradients

```
"Definition of smoothing (spatial filtering) of the gradients" : "comment",
"SPATFILTER" : "0",
"SPAT_FILT_SIZE" : "40",
"SPAT_FILT_1" : "1",
"SPAT_FILT_ITER" : "1",
```

```
Default values are:
SPATFILTER=0
```

One can apply a spatial Gaussian filter to the gradients that acts in horizontal direction (SPATFILTER=1). Have a look at the function 'spat_filt.c' for more information.

6.25 Smoothing the gradients

```
"Definition of smoothing the gradients with a 2D-Gaussian filter" : "comment",
                        "GRAD_FILTER" : "0",
                        "FILT_SIZE_GRAD" : "5",
```

```
Default values are:
GRAD_FILTER=0
```

If GRAD_FILTER = 1 the gradients are smoothed with a 2D median filter after every iterationstep. With FILT_SIZE_GRAD you can set the filter length in gridpoints.

6.26 Limits for the model parameters

```
"Upper and lower limits for model parameters" : "comment",
"VPUPPERLIM" : "3500",
"VPLOWERLIM" : "0",
```

```
"VSUPPERLIM" : "5000",
"VSLOWERLIM" : "0",
"RHOUPPERLIM" : "5000",
"RHOLOWERLIM" : "0",
```

```
Default values are:
VPUPPERLIM=25000.0
VPLOWERLIM=0.0
VSUPPERLIM=25000.0
VSLOWERLIM=0.0
RHOUPPERLIM=25000.0
RHOLOWERLIM=0.0
```

The six limits for the model parameters specify the minimum and maximum values which may be achieved by the inversion. Here, known a priori information can be used. Depending on the choice of the parameter INVMAT1, either vp and vs or lambda and mu is meant.

6.27 Time windowing and damping

```
"Time windowing and damping" : "comment",
"TIMEWIN" : "0",
"PICKS_FILE" : "./picked_times/picks"
"TWLENGTH_PLUS" : "0.01",
"TWLENGTH_MINUS" : "0.01",
"GAMMA" : "100000",
```

```
Default values are:
TIMEWIN=0
```

To apply time windowing in a time series the paramter TIMEWIN must set to 1. A automatic picker routine is not integrated at the moment. The point in time (picked time) for each source must be specified in seperate files. The folder and file name can be set with the parameter PICKS_FILE. The files must be named like this [PICKS_FILE]_sourcenumber.dat. So the number of sources in (SRCREC) must be equal to the number of files. Each file must contain the picked times for every receiver. The parameters TWLENGTH_PLUS and TWLENGTH_MINUS specify the length of the time window after(PLUS) and before (MINUS) the picked time. The unit is seconds. The damping factor GAMMA must be set individually.

6.28 Source wavelet inversion

To remove the contribution of the unknown source time function (STF) from the waveform residuals, it is necessary to design a filter which minimizes the misfit to the field recordings and raw synthetics. This "new" source time function is convolved directly on the observed data. These convolved observed data are used by the calculation of the residuals. Therefore it is not necessary to do a second forward modelling.

```
"Definition of inversion for source time function" : "comment",
"INV_STF" : "0",
"PARA" : "fdlsq:tshift=0.0",
"N_STF" : "10",
"N_STF_START" : "1",
```

```
Default values are:
INV_STF=0
```

A doxygen documentation is published from Thomas Forbriger at http://www.rz.uni-karlsruhe.de/ bi77/txt/cxx/libstfinv/html/index.html.
INV_STF should be switched to 1 if you want to invert for the STF. How to construct parameter strings PARA see doxygen documentation. Examples for the parameter PARA are:

- To select frequency domain least squares (fdlsq) and shift the returned source time function by 0.4s, pass the following parameter string:
  fdlsq:tshift=0.4
- To select fdlsq, apply offset dependent weights and use a power of two to speed up the FFT
  fdlsq:exp=1.4:pow2

N_STF is the increment between the iterationsteps. N_STF_START defines at which iterationstep the inversion for STF should start. This parameter has to be set at least to 1 NOT(!) 0.

Please notice: If you switch additionally frequency filtering during the inversion on (TIME_FILT=1), the parameters N_STF and N_STF_START will be ignored. But the optimal source time function will be inverted for the first iteration and after every change of the frequency range.

6.29 Smoothing the models

```
"Definition of smoothing the models vp and vs" : "comment",
"MODEL_FILTER" : "0",
"FILT_SIZE" : "5",

Default values are:
MODEL_FILTER=0
```

If MODEL_FILTER = 1 vp- and vs-models are smoothed with a 2D median filter after every iterationstep. With FILT_SIZE you can set the filter length in gridpoints.

6.30 Frequency filtering

```
"Frequency filtering during inversion" : "comment",
"TIME_FILT" : "0",
"FC_START" : "10.0",
"FC_END" : "75.0",
"FC_INCR" : "10.0",
"ORDER" : "2",
"ZERO_PHASE" : "0",

Default values are:
TIME_FILT=0
ZERO_PHASE=0
```

The variable TIME_FILT must be set to one to use frequency filtering. The parameter FC_START defines the corner frequency of the Butterworth low pass filter at the beginning of the inversion. The parameter FC_END defines the maximum corner frequency used in the inversion. The parameter FC_INCR controls in which steps the bandwidth is increased during the inversion. The parameter ORDER defines the order of the Butterworth low pass filter. If the variable ZERO_PHASE is set to one a zero phase filter is applied. It is realized by filtering the the traces in both forward and reverse direction with the defined Butterworth low pass filter. Therefore, the effective order of the low pass filter is doubled.

If TIME_FILT is set to one the log file L2_LOG.dat contains a 9th column with the corner frequency in Hz used in the iteration step.

With the parameter PRO (see 6.21) one has to adjust the criterion that defines at which points the bandwidth of the signals are increased.

6.31 Trace killing

```
"Trace killing" : "comment",
"TRKILL" : "0",
"TRKILL_FILE" : "./trace_kill/trace_kill.dat",


Default values are:
TRKILL=0
```

For not using all traces, the parameter TRKILL is introduced. If TRKILL is set to 1, trace killing is in use. The necessary file can be selected with the parameter TRKILL_FILE. The file should include a kill table, where the rows have the same lengths as the number of receivers and the columns reflects the number of sources. Now, a 1 (ONE) means, YES, please kill the trace. The trace is NOT used, it should be killed. A 0 (ZERO) means, NO, this trace should NOT be killed.

Chapter 7

Example 2 - the elastic Marmousi2 model

Developed in the 1990s by the French Petroleum Institute (IFP) (Versteeg [1994]) the Marmousi model is a widely used test problem for seismic imaging techniques. Beside the original acoustic version of the model an elastic version was developed by Martin et al. [2006]. This model consists of parts with simple (approximately 1D) and complex geological situations. In the following two sections the performance of the FWT code will be tested for the complex and the simple part of the Marmousi2 model, respectively.

7.1 The complex Marmousi2 model

The Marmousi2 model (Fig. 7.1) consists of a 500 m thick water layer above an elastic subseafloor model. The sediment model is very simple near the left and right boundaries but rather complex in the centre. At both sides, the subseafloor is approximately horizontally layered, while steep thrust faults are disturbing the layers in the centre of the model. Embedded in the thrust fault system and layers are small hydrocarbon reservoirs (figure 7.1, Martin et al. [2006]).

- One shallow gas sand in a simple structural area (A).
- One relatively shallow oil sand in a structural simple area (B).
- Four faulted trap gas sands at varying depths (C1,C2,C3,C4).
- Two faulted trap oil sands at medium to deep depths (D1,D2).
- One deep oil and gas sand anticlinal trap (E1,E2).
- Water wet sand.

The deeper parts of the model consist of salt and reef structures. The thrust fault system and the reef structures are not easy to resolve by conventional first arrival tomography, so it is an ideal test model for the FWT. Due to computational restrictions the original Marmousi-II model could not be used, because the very low S-wave velocities in the sediments would require a too small spatial sampling of the model. Therefore new S-wave velocities are calculated using the empirical scaling relation for hard rocks. Additionally the size of the Marmousi2 model is reduced from 17×3.5 km to 10×3.48 km (figure 7.2).

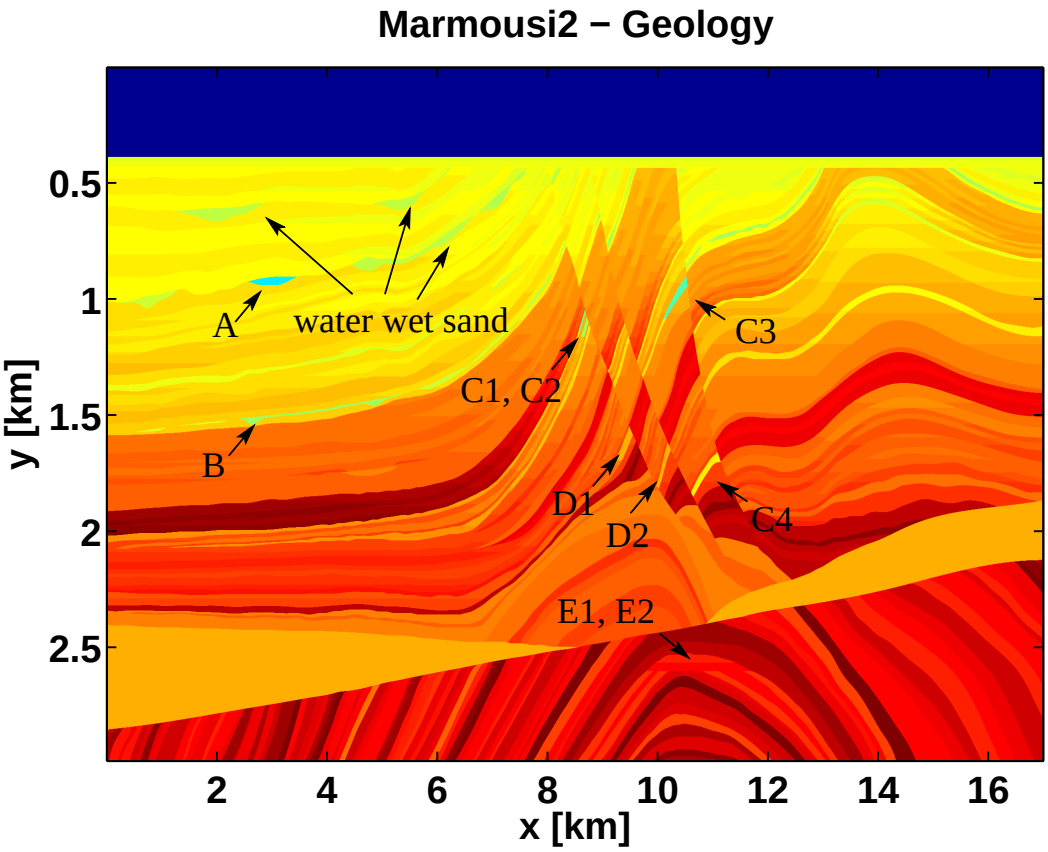


Figure 7.1: Marmousi2 model - geology.

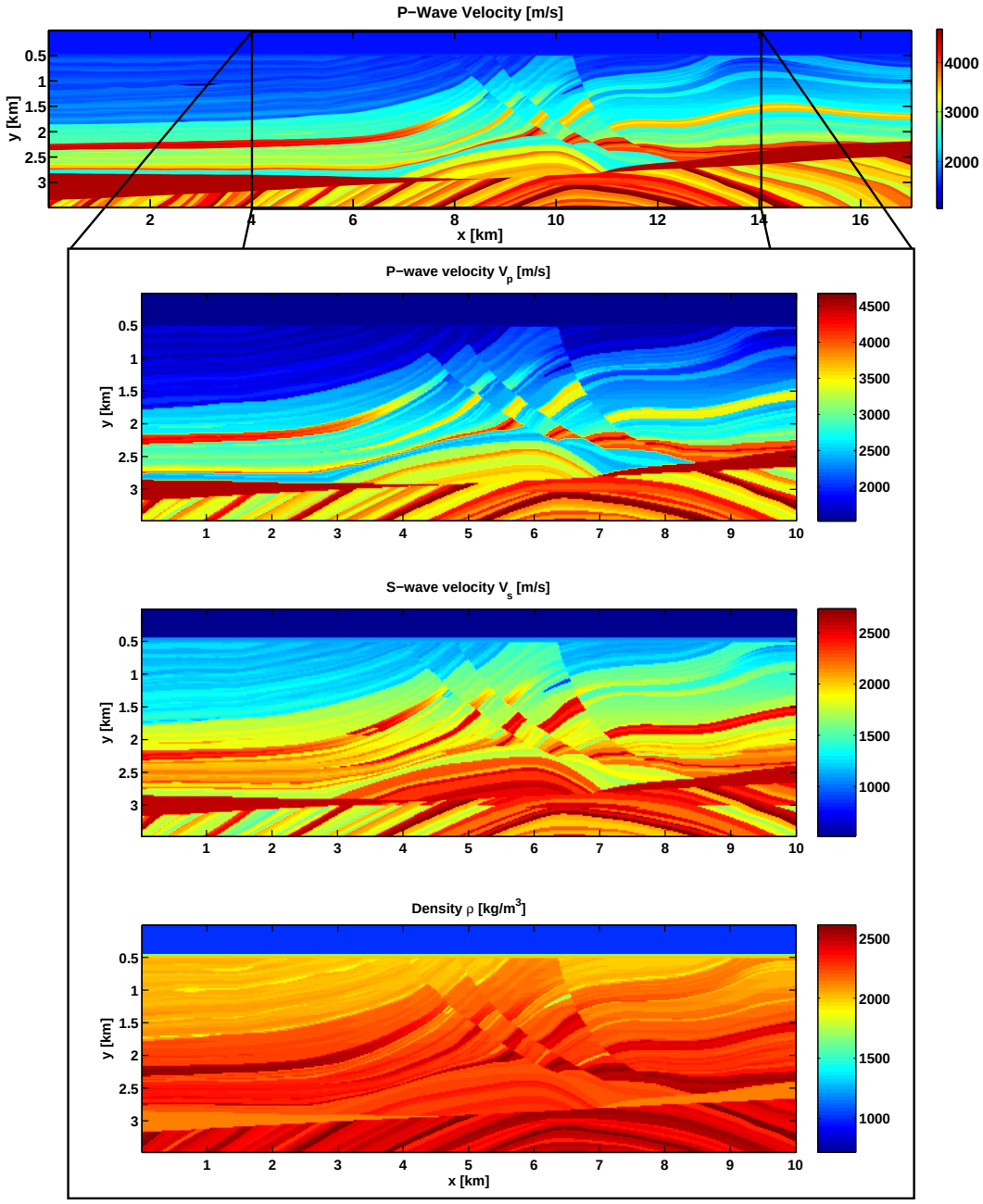


Figure 7.2: The reduced complex Marmousi2 model used for the elastic FWT.

7.1.1 Acquisition geometry and FD model

The acquisition geometry consists of a fixed streamer located 40 m below the free surface in the water layer. The streamer contains 400 two component geophones recording the displacements $\mathrm{u_i}$. For the synthetic dataset 100 airgun shots are recorded. The sources are located at the same depth as the receivers. The source signature is a 10 Hz Ricker wavelet. The model has the dimensions $10 \times 3.48\ \mathrm{km}$. Using an 8th order spatial FD operator the model can be discretized with 500×174 gridpoints in x- and y-direction with a spatial gridpoint distance of 20.0 m. The time is discretized using DT = 2.7 ms, thus for a recording time of T = 6.0 s 2222 time steps are needed.

7.1.2 Elastic wave propagation in the complex Marmousi model

Fig. 7.3 shows the development of the pressure wavefield excited by shot 50 for the central part of the complex elastic Marmousi2 model at 6 different time steps. The P wave is traveling from the source through the water column (T=100.0 ms) and is reflected at the seafloor (T=400.0 ms). In the elastic subseafloor medium the wavefield becomes very complex. The layers in the steep thrust fault system produce numerous reflections and internal multiples (T=600.0 ms). Additionally strong diffracted waves are generated at the sharp corners of the thrust faults between the disturbed high velocity sediment blocks within the thrust faults and the surrounding low velocity sediments. At the free surface strong multiple reflections occur (T=800.0 ms). The wavefront of the direct wave is quite deformed due to strong velocity contrasts within the thrust fault system. After 1500 ms nearly all kinds of waves which can be found in the literature are present: Reflections, refractions, diffractions, (internal) multiples or interface waves. The trapped gas sand reservoirs C1, C2 and C3 produce strong reflections and mode conversions. This complexity is also visible in the seismic section, recorded by the streamer in the water column. As an example Fig. 7.9, (f) shows the seismic section of the y-component for shot 50. Beside the direct wave and a strong reflection from the seafloor numerous small reflection events from the thrust fault system are dominating the seismic section.

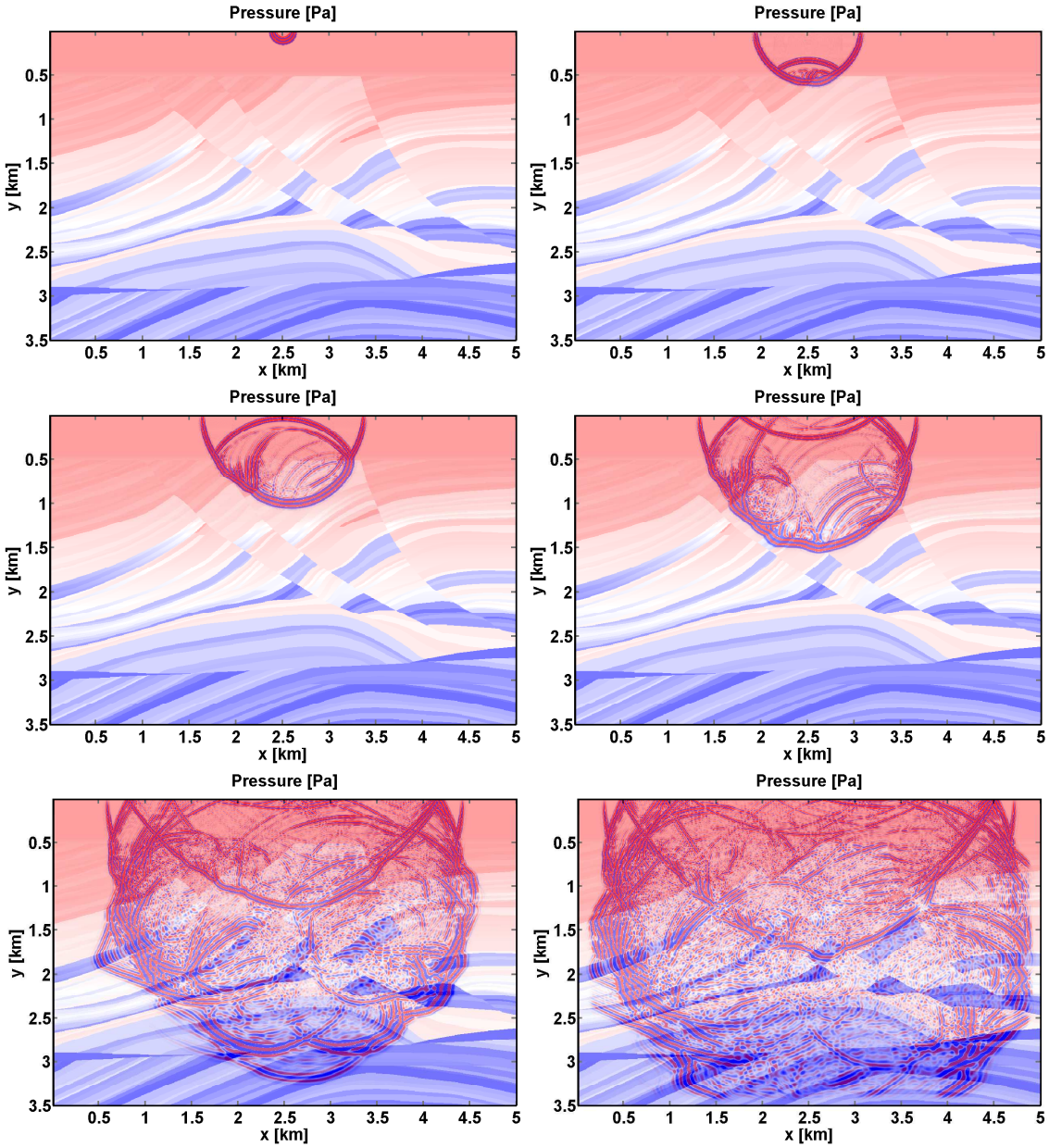


Figure 7.3: Pressure wavefield excited by shot 50 for the elastic Marmousi2 model at 6 different time steps .

7.1.3 FWT of the complex Marmousi model

Due to the results of the last section, I choose the seismic velocities as model parameters for the inversion. To generate a starting model which describes the long wavelength part of the material parameters correctly the true model $\mathbf{m} = [\mathrm{V_p},\ \mathrm{V_s},\ \rho]$ was filtered using a spatial 2D-Gaussian filter

$$\mathbf{m}_{\mathrm{smooth}}(\mathrm{x},\mathrm{y}) = \frac{1}{2\pi\lambda_\mathrm{c}^2}\int_{-\lambda_\mathrm{c}}^{\lambda_\mathrm{c}}\int_{-\lambda_\mathrm{c}}^{\lambda_\mathrm{c}} \mathrm{dx'dy'}\mathbf{m}(\mathrm{x}-\mathrm{x'},\mathrm{y}-\mathrm{y'})\exp\left(-\frac{(\mathrm{x}-\mathrm{x'})^2+(\mathrm{y}-\mathrm{y'})^2)}{2\lambda_\mathrm{c}^2}\right) \tag{7.1}$$

with a correlation length $\lambda_\mathrm{c} = 200.0\ \mathrm{m}$. As a result all the small scale structures vanished and only the large scale structures are present (Fig. 7.4).

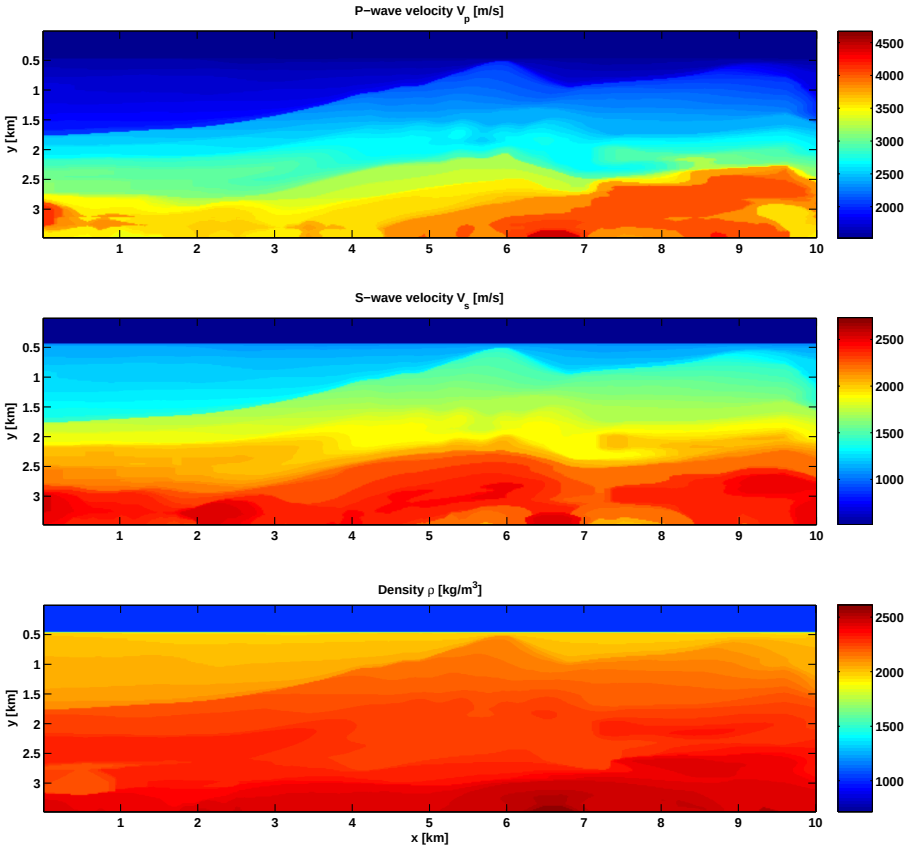


Figure 7.4: Starting models for the Marmousi-II model.

As in the case of the spherical low velocity anomaly the application of a preconditioning operator is vital to suppress the large gradient values near the source and receiver positions. Additionally strong artefacts are present near the free surface (figure 7.6, top) which are a few orders of magnitude larger than the gradient of the material parameters. This problem is also known in case of the acoustic inversion problem (Ben-Hadj Ali et al. [2008]). To suppress these effects a spatial preconditioning operator is applied on the gradient, which is defined as (figure 7.5)

$$\mathrm{P} = \begin{cases} 0 & \text{if } \mathrm{y}_0 = 0.0 \text{ m} \leq \mathrm{y} \leq \mathrm{y}_{\mathrm{gradt1}} = 380.0 \text{ m} \\ \exp(-\frac{1}{2}(\mathrm{a}\frac{\mathrm{y}-\mathrm{y}_{\mathrm{gradt1}}-\Delta\mathrm{l}}{\Delta\mathrm{l}/2})^2) & \text{if } \mathrm{y}_{\mathrm{gradt1}} = 380.0 \text{ m} \leq \mathrm{y} \leq \mathrm{y}_{\mathrm{gradt2}} = 480.0 \text{ m} \\ \frac{\mathrm{y}}{\mathrm{y}_{\mathrm{gradt2}}} & \text{if } \mathrm{y}_{\mathrm{gradt2}} = 480.0 \text{ m} < \mathrm{y} \end{cases} \tag{7.2}$$

with $\mathrm{a} = 3.0$, $\Delta\mathrm{l} = 100.0$ m.

The preconditioning operator sets the gradient near the free surface and the sources/receivers to zero. In a transition

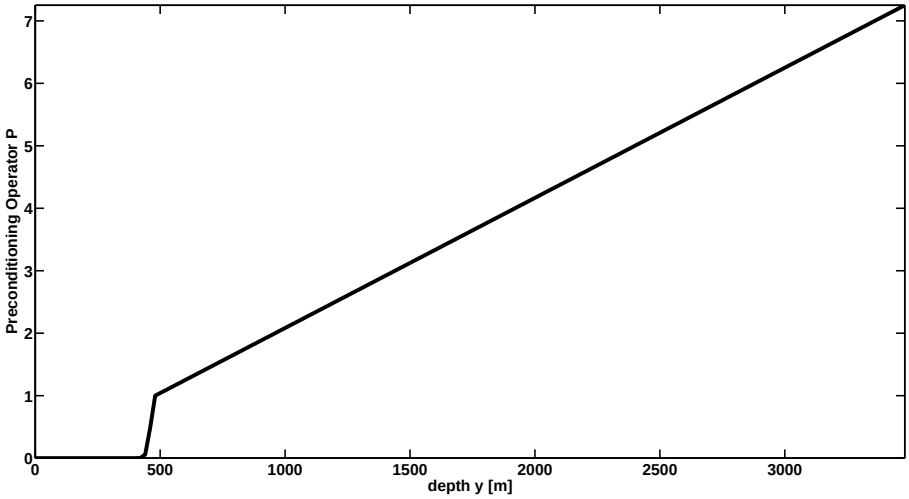


Figure 7.5: The preconditioning operator for the reflection geometry.

zone between 380.0 m and 480.0 m depth a Gaussian taper is applied. Beyond a depth of 480 m the operator scales the gradient linear with depth. This is a very crude correction for the amplitude loss in larger depths due to geometrical spreading and reflections in the upper parts of the model. Before the application of the preconditioning operator no subsurface structures are visible at all figure 7.6 (top), while strong reflectors are visible after its application.

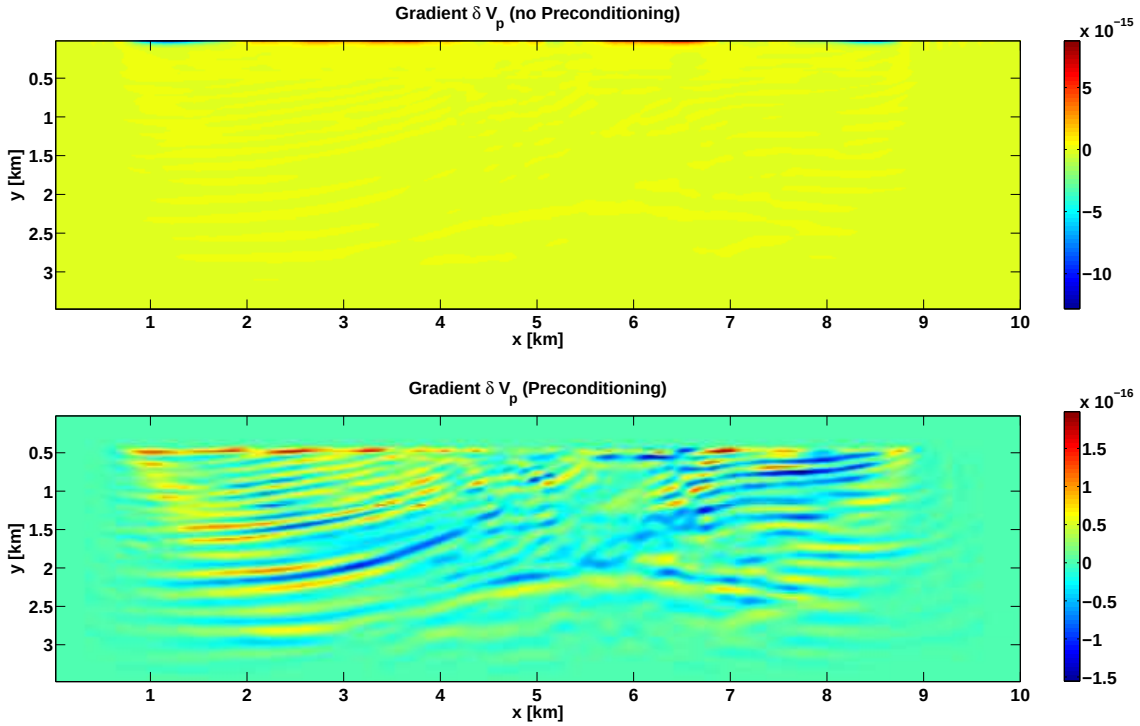


Figure 7.6: The influence of the preconditioning operator P for the elastic Marmousi2 model. The Gradient $\delta \mathrm{V_p}$ (iteration no. 1) before (top) and after the application of the preconditioning operator (bottom).

To achieve a smooth transition from the long wavelength starting model to the inversion result with short wavelength structures the application of a frequency filter with variable bandwidth on the data residuals $\delta \mathbf{u}$ is vital, to avoid the convergence into a local minimum. In this case the inversion is separated in two parts. In part I only frequencies below 10 Hz are inverted, while in part II the full spectral content up to 20 Hz is inverted. The inversion results after 350 iterations are shown in Fig. 7.7. Additionally depth profiles at $\mathrm{x_{p1}} = 3.5\ \mathrm{km}$ and $\mathrm{x_{p2}} = 6.4\ \mathrm{km}$ of the starting model and inversion result are compared with the true model in Fig. 7.8. The results contain a lot of small details. All fine layers which are completely absent in the starting model could be resolved. The thrust faults and the reef structures in the deeper part of the model are imaged also very well. Despite the deep hydro carbon reservoirs E1 and E2, the oil and gas reservoirs C1, C2, C3, C4, D1 and D2 are all clearly visible.

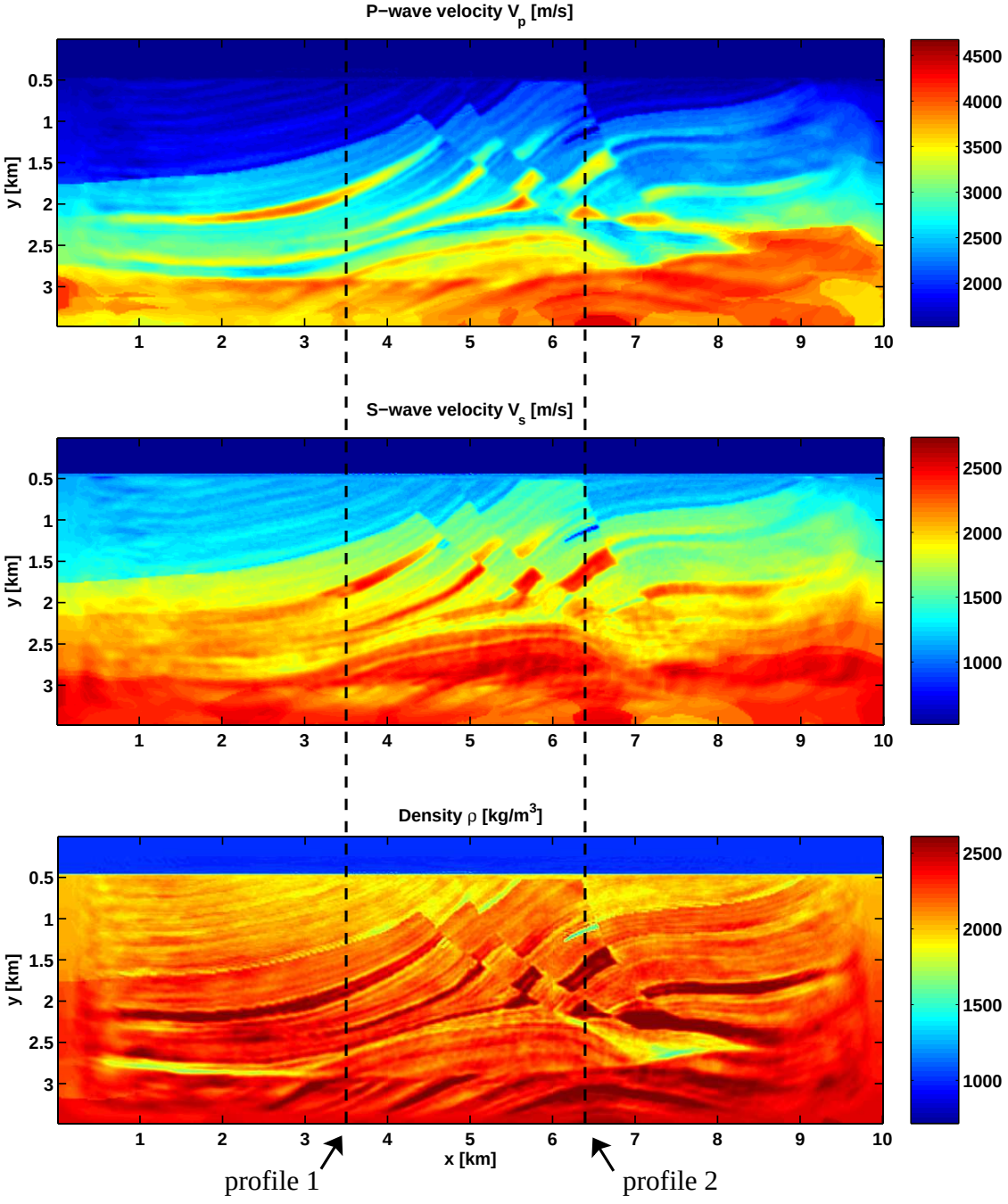


Figure 7.7: Results of the elastic FWT for the Marmousi-II model. The dashed lines denote the positions of the depth profiles 1 and 2 shown in Fig. 7.8.

It is quite surprising, that the shear wave velocity model could also be resolved very well, even though only streamer data and therefore mainly P-wave information is used. Even the density, a parameter which can be hardly estimated from seismic data, could be recovered from the seismic wavefield. Keep in mind though, that the density image is based not only density information, but contains also $\mathrm{V_p}$ and $\mathrm{V_s}$ information due to the ambiguity investigated by the CTS test problem [Köhn, 2011, chapter 5.2]. The depth profiles show the strong deviation of the starting model from the true Marmousi model, in some parts up to $\pm 500 - 1000\ \mathrm{m/s}$ for the velocity models and $\pm 250 - 500\ \mathrm{kg/m^3}$ for the density models. Even though the FWT could reconstruct the velocity and density structures quantitatively fairly well. In Fig. 7.9 the seismic section of shot 50 is plotted for the starting model (a), the inversion result (b) the true model (c), the initial residuals (d), the final residuals (e) and the evolution of the residual energy (f). Notice the good fit of the first arrivals for the starting model, but the lack of small details beyond the first arrivals. Only the direct wave, the reflection from the ocean bottom and a few multiples are present. The inversion result fits most of the phases and amplitudes of the later small scale reflections from within the thrust fault system and therefore the data residuals are very small. The misfit function decreases smoothly for the first 20 Iteration steps, but exhibits strong fluctuations for the later iterations, due to the strong nonlinear character of the inversion problem for this complex geology. The performance of the FWT code was benchmarked on an Altix 4700 using the Marmousi2 model. Fig.7.10 shows the computation time (top) and the memory requirements for up to 50 CPUs (bottom). On a single CPU the elastic timedomain FWT is not efficient at the moment. For 150 iteration steps a total calculation time of 85 days and about 10 GB of RAM would be required. When using 50 CPUs the computation time is reduced to roughly 2 days. Note the linear speedup of the FWT code due to the optimized parallelization of the forward modelling FD code.
In conclusion the results look very impressive, but you should not forget that the starting model for the seismic velocities and density are unrealistically accurate and are not easy to derive from the seismic data for such a complex model.

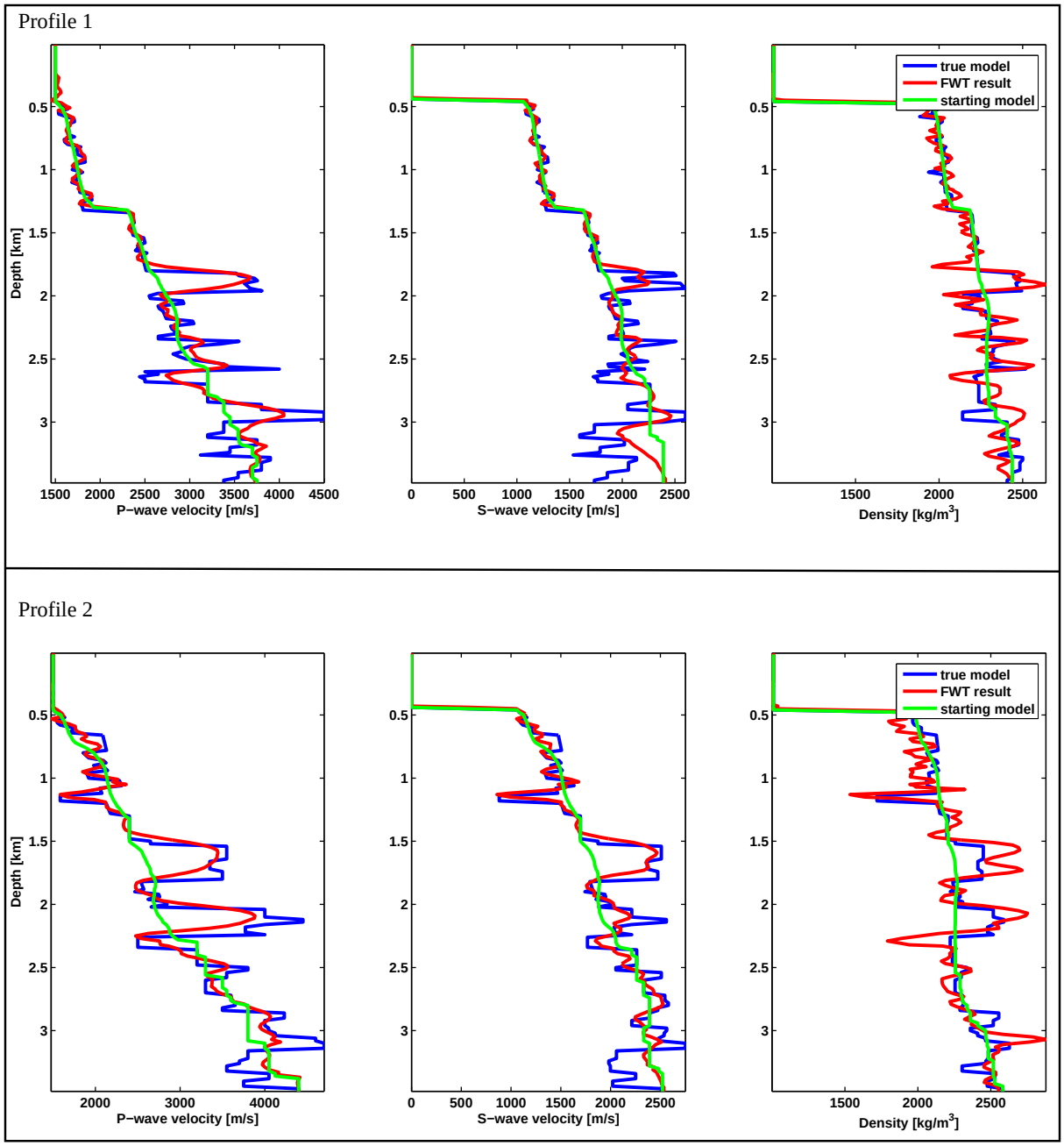


Figure 7.8: Depth profiles at $x_{p1} = 3.5$ km (top) and $x_{p2} = 6.4$ km (bottom) of the starting model and FWT result are compared with the true model for the Marmousi-II model: P-wave velocity (left), S-wave velocity (center) and density (right).

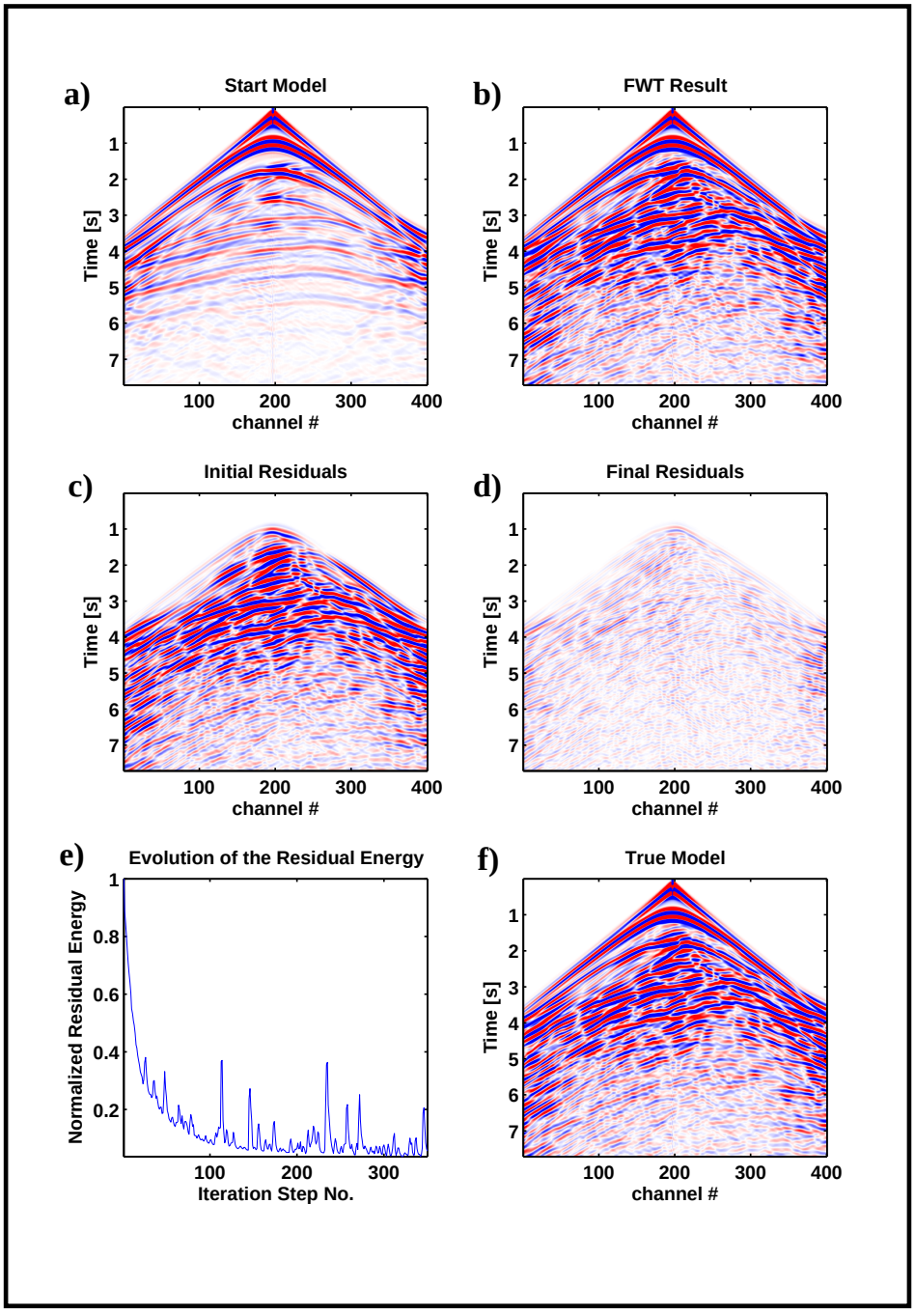


Figure 7.9: Seismic sections (shot 50, y-component) for the Marmousi-II model. (a) starting model, (b) FWT result, (c) initial residuals, (d) final residuals , (f) true model and (e) evolution of the residual energy.

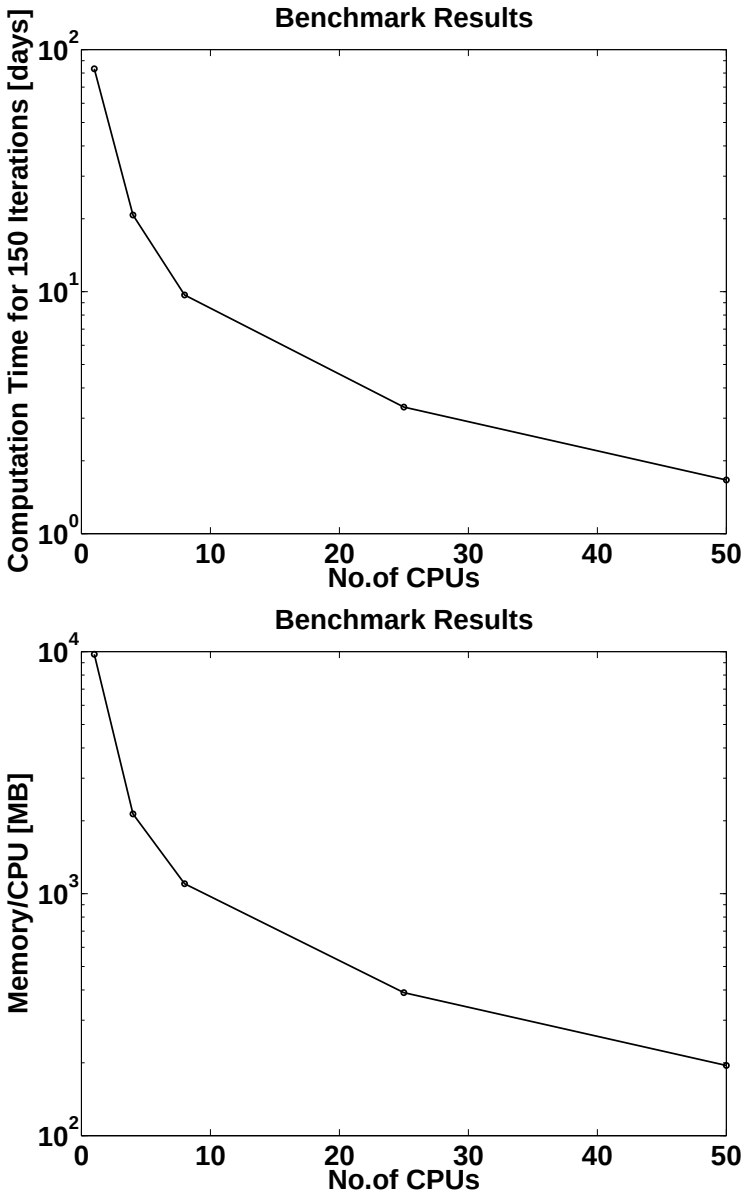


Figure 7.10: Benchmark results for the Marmousi2 model. The absolute calculation times (top) and the memory requirements (bottom) for up to 50 CPUs on an Altix 4700.

Chapter 8

Example 3 - inversion of ultrasonic data

8.1 Homogenous plexiglas bloc model

The second example consists of real data which was recorded at a Plexiglas bloc with size of 10*15*20 cm. Using a simple regression a Rayleigh-wave-velocity of 1277.2 m/s could be estimated. A waveform inversion with DENISE can be achieved in XX steps. The necessary MATLAB scripts are located in the */mfiles/US* directory.

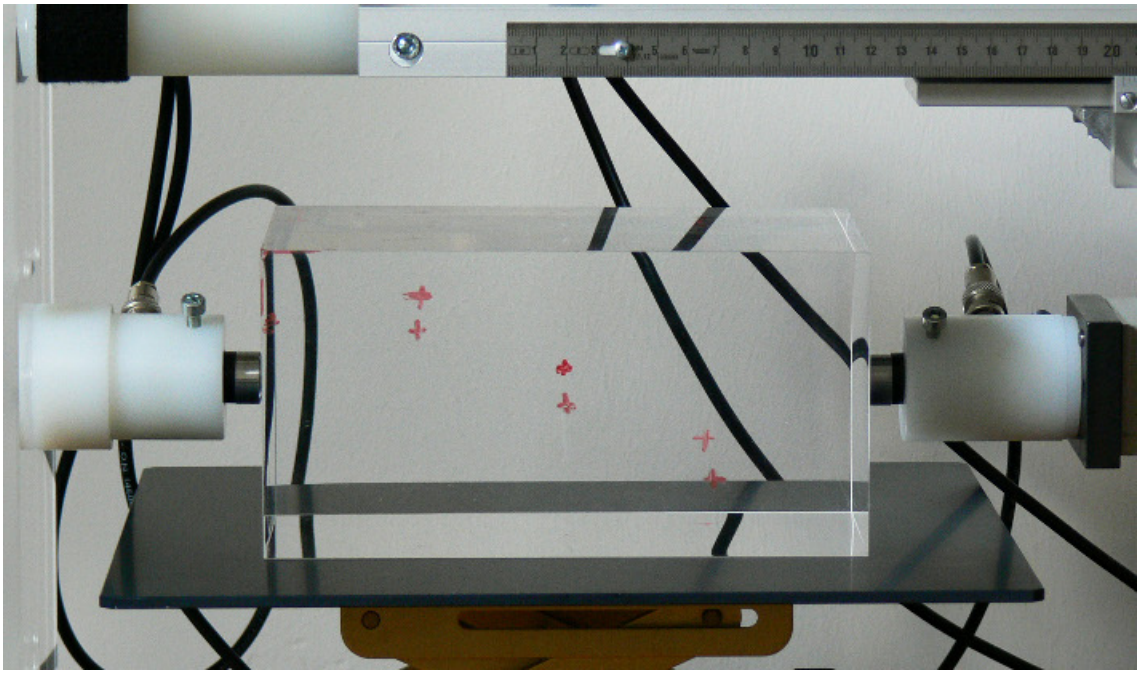


Figure 8.1: Photo of the homogenous plexiglas bloc with ultrasonic acquisition geometry.

1. First we have to find some initial guess for the elastic parameters of plexiglas. The parameters in Zerwer et al. [2000] seem to be in good agreement with the measured Rayleigh wave velocity: $\mathrm{Vp} = 2700\ \mathrm{m/s}$, $\mathrm{Vs} = 1370\ \mathrm{m/s}$ and $\rho = 1190\ \mathrm{kg/m^3}$ leads to a Rayleigh wave velocity of $\mathrm{VR} = 1280\ \mathrm{m/s}$. Assuming a maximum frequency $\mathrm{f_{max}} = 600\ \mathrm{kHz}$ and using the grid disperson-, Courant criterion and the extension of the acquisition geometry the following FD-model parameters are required:

   - DH = 2.0e-4 m
   - DT = 5.2e-8 s
   - NX = 860 gridpoints, NY = 200 gridpoints
   - NT = 3077 time steps

   The necessary files for the input parameters DENISE_US.inp, source_ricker.dat, source_spike.dat and source_wavelet.dat, receiver_US.dat and model definition half_space.c are located in the directory DENISE/genmod/US. Copy these files to the DENISE/par and DENISE/src directories, respectively. Set

```
QUELLART=1
SOURCE_FILE = ./source/source_ricker.dat
SNAP=3
INVMAT=10
```

   in the parameter file *DENISE_US.inp* and start a first test model run:

```
-bash-2.05b$:~/DENISE/par> mpirun -np 4 ../bin/denise DENISE_US.inp
```

   Merge the resulting wavefield snapshots with

```
-bash-2.05b$:~/DENISE/par> snapmerge ../bin/denise < DENISE_US.inp
```

   Copy the file snap/waveform_forward.bin.rot to the snap-directory in mfiles/US. With the MATLAB-script US_snap you can visualize the propagation of the surface wave. The script is self-explanatory. Set

```
14 % name of the snap file
15 file_div='snap/waveform_forward.bin.rot';
[...]
20 % optimize clipping values
21 caxis_value_vp1=-5e-1;
22 caxis_value_vp2=5e-1;
[...]
24 % define first and last frame to display
25 firstframe=1;
26 lastframe=50;
[...]
28 % gridsize and grid spacing (as specified in parameter-file)
29 NX1glob=1; NX2glob=860;
30 NY1glob=1; NY2glob=200;
31 IDX=1; IDY=1;
32 dh_glob=2e-4;
[...]
34 % time increment for snapshots:
35 TSNAPINC=3e-6; TSNAP1=3e-6;
```

   and run the script. A few representive snapshots of the Rayleighwave propagation are shown in figure 8.2

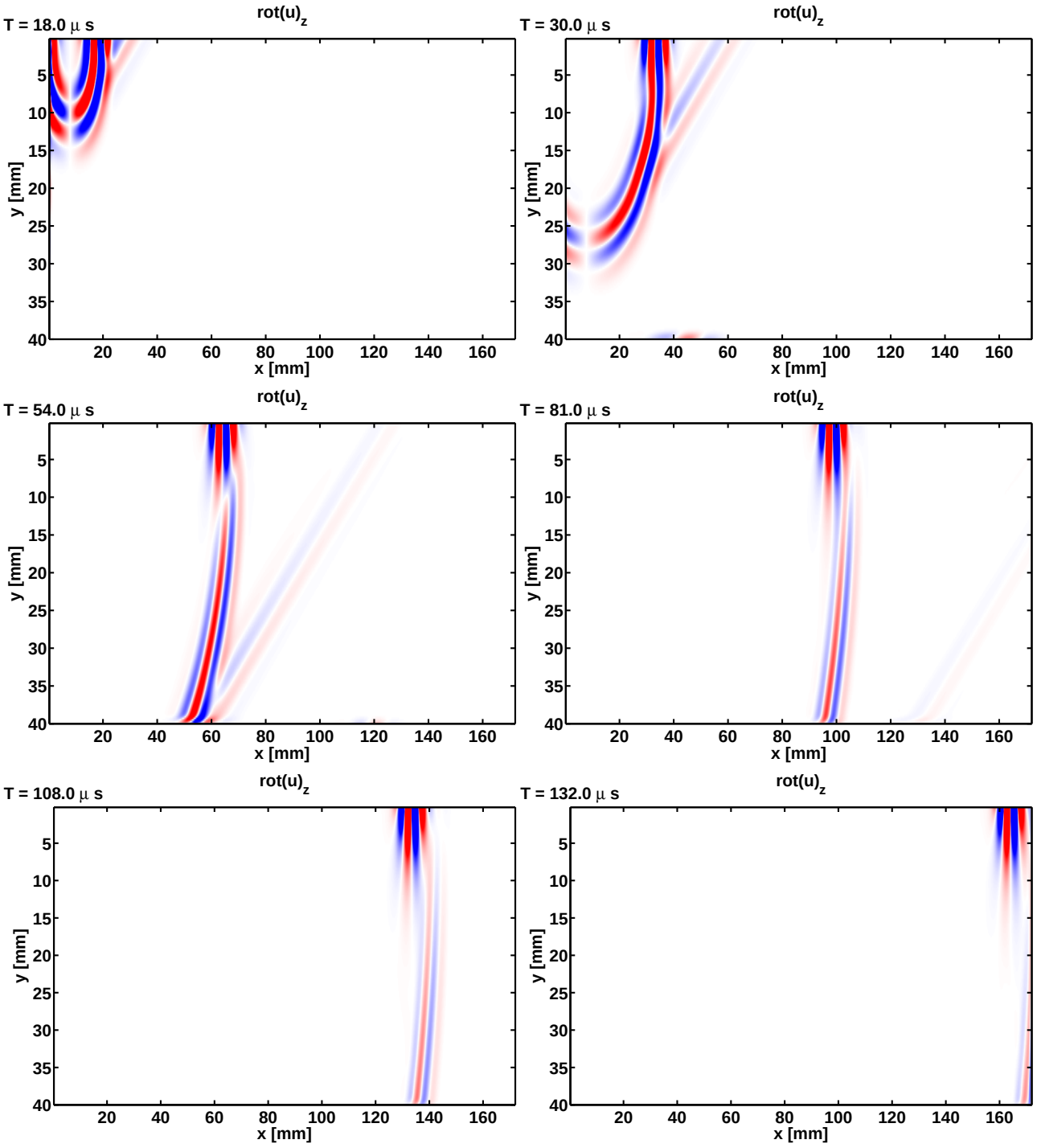


Figure 8.2: A few snapshots of the Rayleighwave propagating through the homogenous Plexiglas block.

2. In the next step we take a look at the raw data and convert the ASCII data to SU format. Open the MATLAB-script ASCII2SU.m. Define the location of the plexiglas data:

```
83 file_name = ['plexiglas_profil_data/',int2str(i),'mm.txt'];
```

Each trace of the ultrasonic data is located in a file $< \mathbf{offset} > \mathbf{mm.txt}$, the acquistion geometry is defined in $\mathbf{readme.txt}$. The range of the offset values, the increment and the sample interval:

```
63 ntr1=62;          % minimum offset
64 ntr2=162;         % maximum offset
65 dntr1=10;         % offset interval
...
69 DT=1.0e-7;        % sampling interval of the US data
```

Activate the interpolation of the data and define the sampling interval and number of time steps in DENISE:

```
41 INTERP=1;
42 DT1=5.2e-8;     % sampling rate for DENISE
43 nt_DENISE=3462; % number of time steps in DENISE
```

Define a time delay, which makes the later source wavelet inversion much easier:

```
23 DELAY=1;
24 delay=2.8e-5;   % time delay for real data
```

and finally activate the output of the SU file and enter a name for the SU file:

```
55 WRITE=1;
56 % name of output SU file
57 imfile=['plexiglas_profil_data/DENISE_plexiglas_y.su.shot1'];
```

After running the script you get the following seismic section:

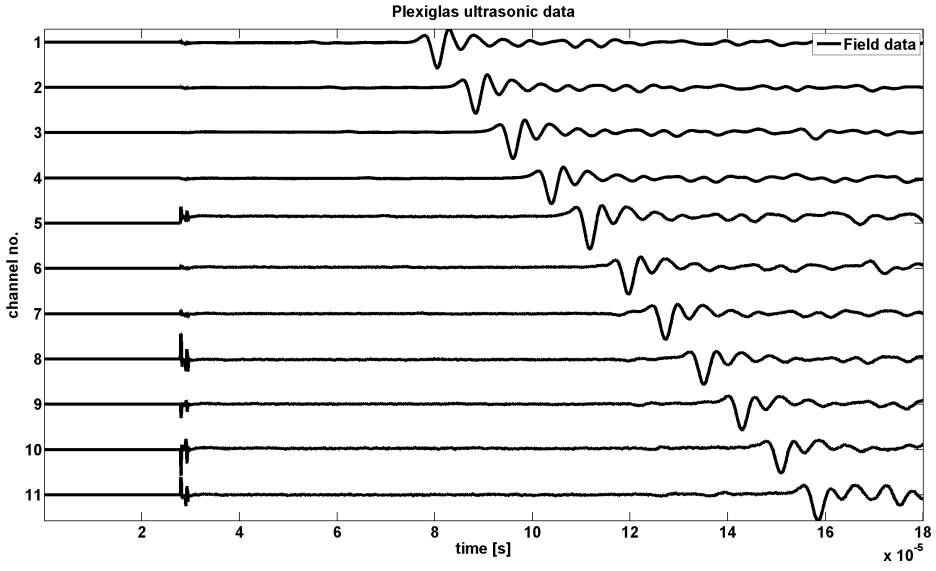


Figure 8.3: Elastic waveform inversion of ultrasonic data - step 1 (raw data).

3. The data looks a bit noisy, therefore activate the bandpass frequency filter, where f1, f2, f3 and f4 define the corner frequencies:

```
47 FILTER=1;
48 f1=35000;
49 f2=40000;
50 f3=450000;
51 f4=500000;
```

We want to restrict the inversion to the wavelet of the first arrival. Therefore activate the time window:

```
28 TWIN=1;
29 twinc=5.36e-5+delay; % center of time window
30 twinp=1e-5;          % twinc + twinp
31 twinm=1e-5;          % twinc - twinm
32 ad=1e12;             % damping parameter
33 vnmo=1280;           % velocity of the Rayleigh wave
```

The parameter twinc should be approximately at the center of the first arrival wavelet. twinp and twinm define the length of the time window, ad the damping parameter and vnmo the normal moveout of the Rayleigh wave. Optimize these parameters until the window is well defined. For each trace the value of twinc will be written to

```
278 imfile=['picks_1.dat'];
```

An optimized seismic section is shown in figure 8.4. The yellow dots denote the time twinc, the blue dots the time twinc+twinp and the red dots the time twinc-twinp, respectively.

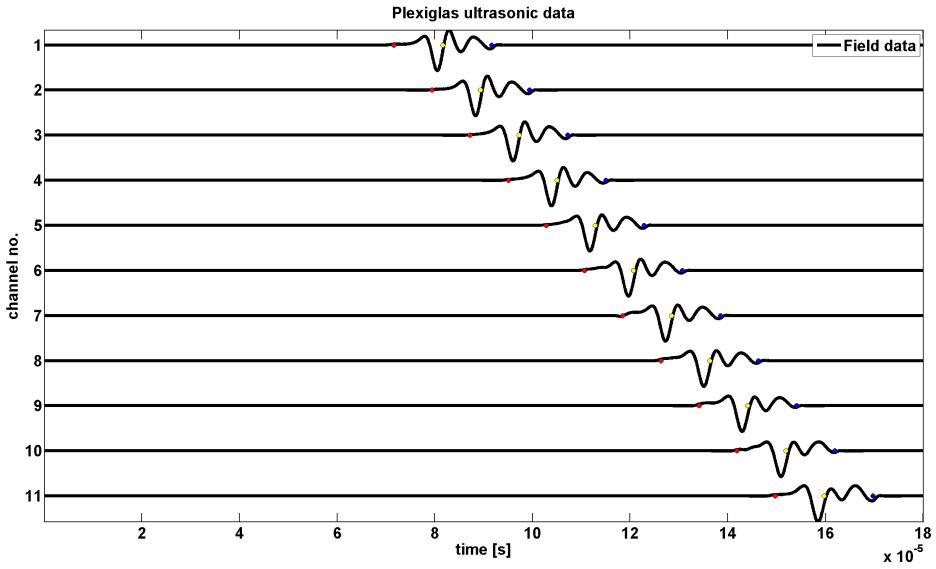


Figure 8.4: Elastic waveform inversion of ultrasonic data - step 2 (optimized seismic section with time window).

4. Copy the generated SU-file *DENISE_plexiglas_y.su.shot1* to the data directory *DENISE/par/su/plexiglas* and the file with the definition of the time window *picks_1.dat* to *DENISE/par/picked_times*.

5. Define

```
QUELLART=6
SOURCE_FILE = ./source/source_spike.dat
SNAP=0
INVMAT=10
TWIN=0
INV_STF=0
```

   in the parameter file *DENISE_US.inp*, set the time delay td in the source-file *source_spike.dat* to 0.0 s and run a forward model.

6. Copy su/DENISE_US_y.su.shot1.it1 to the FD_seis directory and open the Matlab script Plot_US_Residuals.m. Be sure that the parameters delay, twinc, twinp, twinm, ad, vnmo, f1, f2, f3, f4, ntr, ntr1, ntr2, dntr1, DT1 and in_data are equal to the definitions in ASCII2SU.m. Turn the time window off (TWIN=0). After running the script you should get the following seismic sections. On the left side a comparison of the seismic section between the model data (red) and the field data (black) and on the right the data residuals:

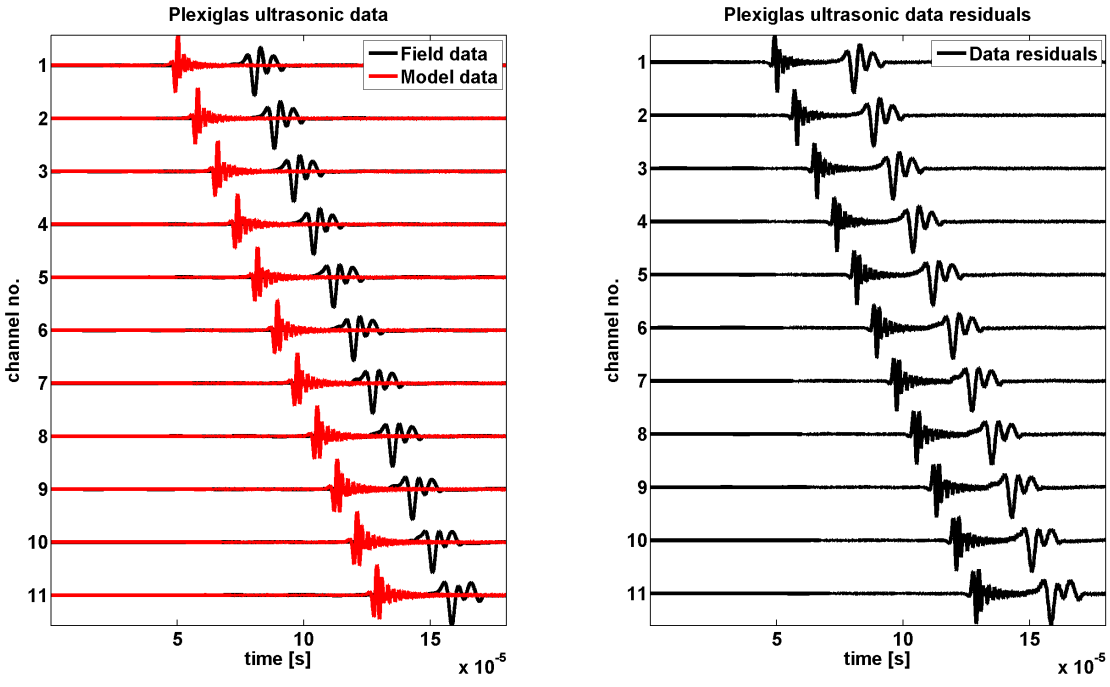


Figure 8.5: Elastic waveform inversion of ultrasonic data - step 5 (first model result).

7. Zoom into the first trace of the data comparison and pick time values of the measured t1 and modelled wavelets t2, respectively (figure 8.6). From this values calculate the time delay td between the wavelets. In this example, t1=8.05e-5 s, t2=4.93e-5 s and td=t1-t2=3.12e-5 s.

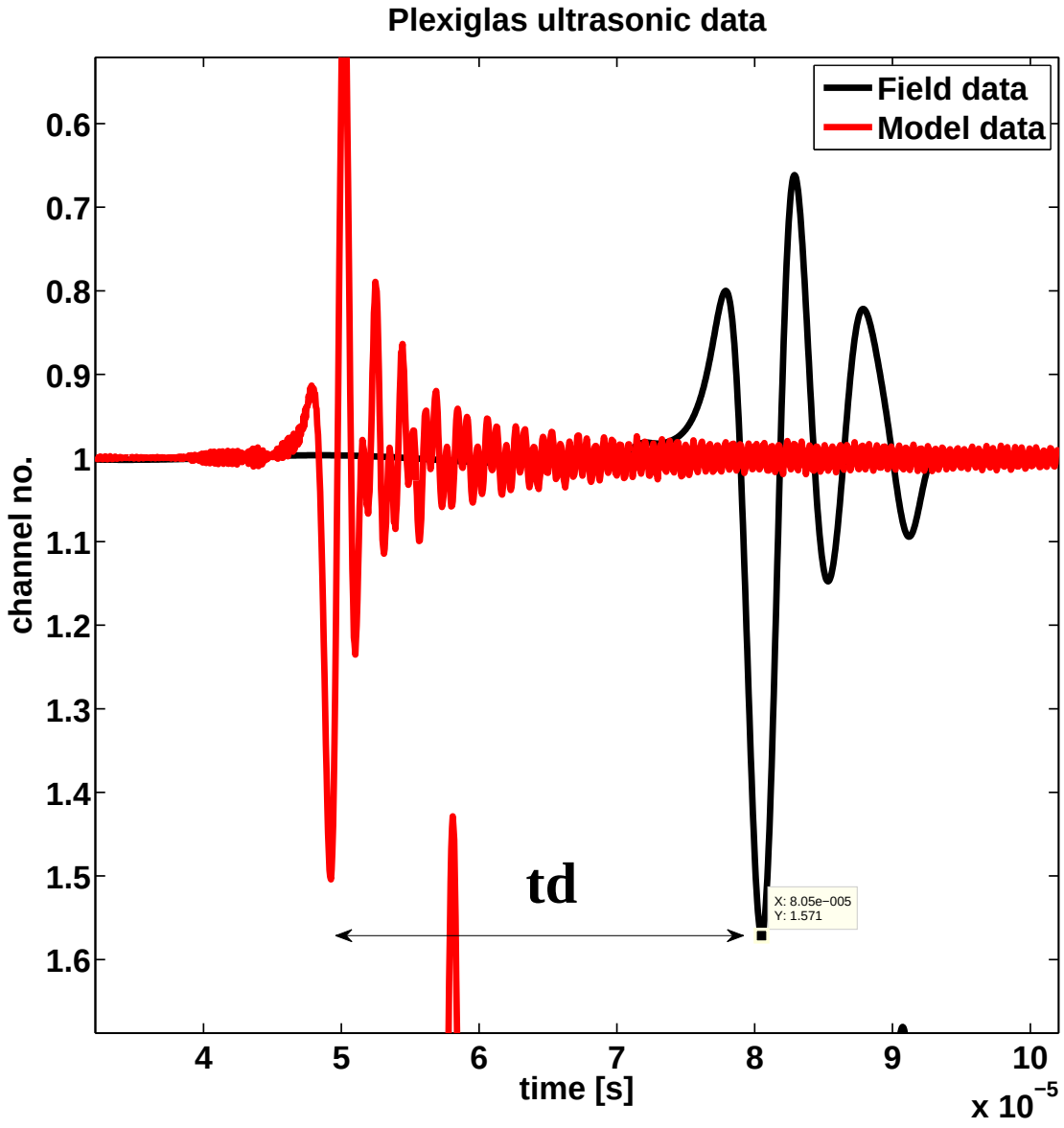


Figure 8.6: Elastic waveform inversion of ultrasonic data - step 6 (measure time delay).

8. Change the time delay td in source_spike.dat to the measured value, run a new model, copy the resulting DENISE_US_y.su.shot1.it1 file to the FD_seis directory again and check if there is still a significant time delay present. Optimize it further until the wavelets approximately fit (figure 8.7).

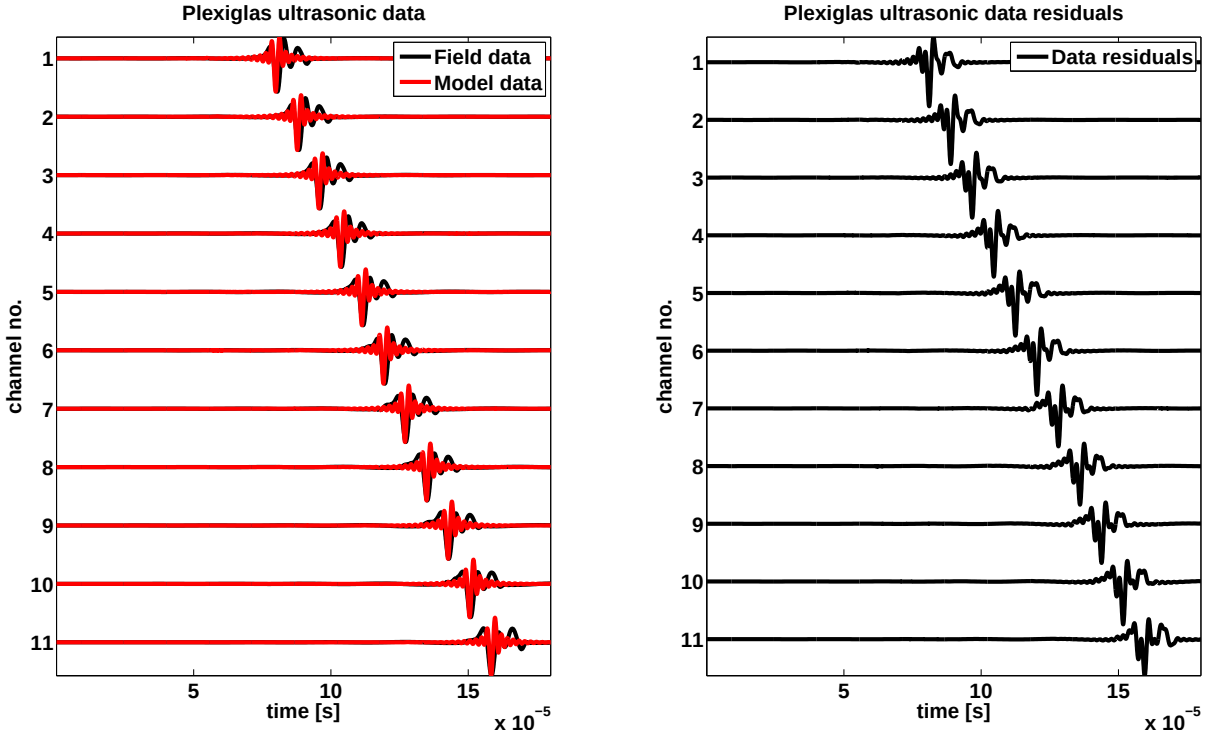


Figure 8.7: Elastic waveform inversion of ultrasonic data - step 7 (optimum wavelet fit).

9. After fitting the arrival times of the wavelets approximately we proceed with the source wavelet inversion. Copy the initial source wavelet from start/half_space_source_signal.0.su.shot1 to US/wavelet. Open the matlab script est_source_wavelet.m and check if the parameters delay, twinc, twinp, twinm, ad, vnmo, f1, f2, f3, f4, ntr, ntr1, ntr2, dntr1, DT1, in_data and in_model are consistent with those in the other two matlab scripts and additionally define in_source. Set the Marquardt-Levenber damping factor epsilon=0.0 and run the script. With no regularization the data fit looks nearly perfect (figure 8.8), but the resulting source wavelet looks very unrealistic and contains a lot of spikes. When increasing the damping factor epsilon (f.e. to epsilon=1e1) the solutions are damped and become more plausible, while the data misfit increases (figure 8.9). Optimize epsilon until you find a good compromise between plausible source wavelet and minimum data misfit.

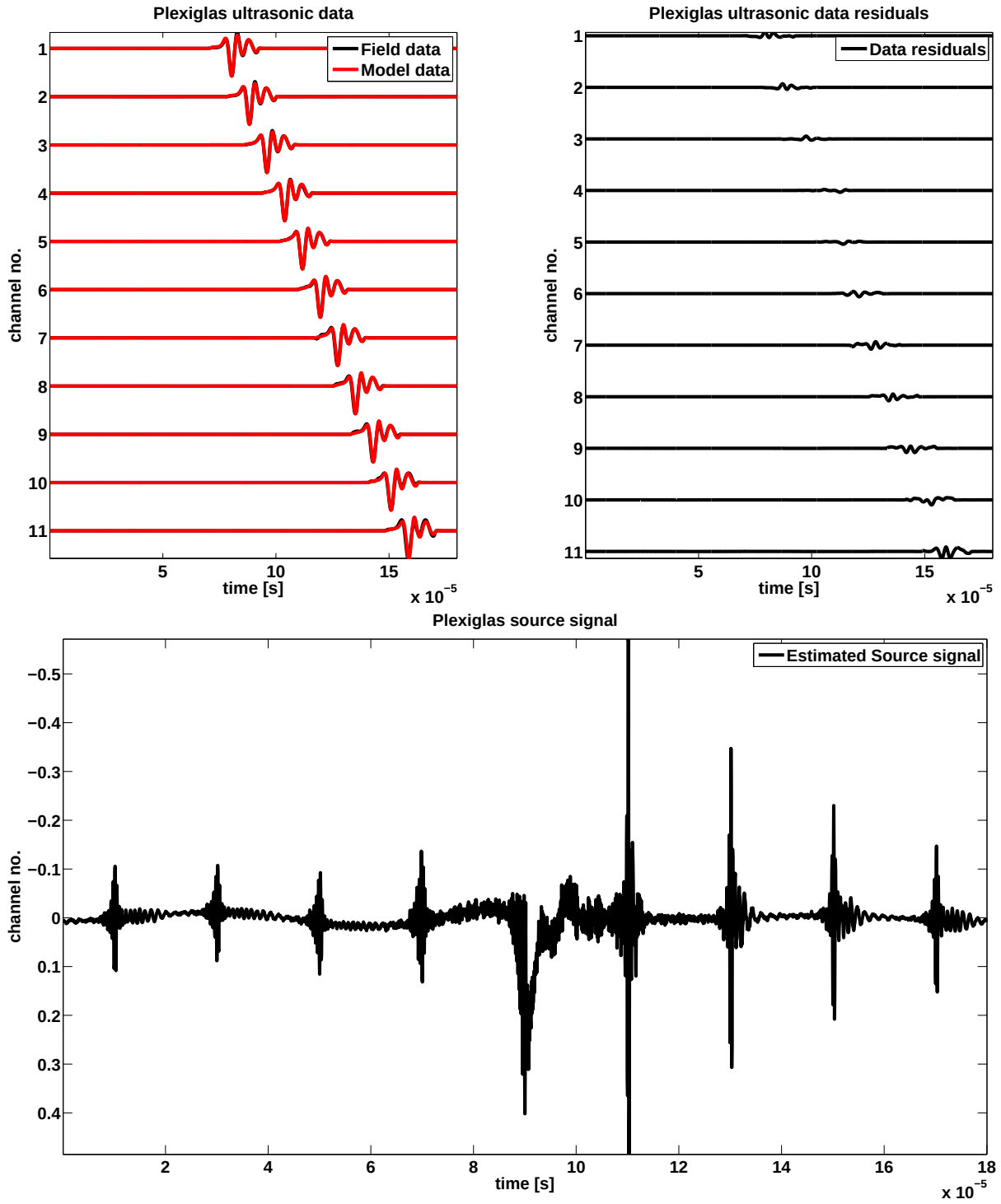


Figure 8.8: Elastic waveform inversion of ultrasonic data - step 8. Resulting seismic sections for the estimated source wavelet (top) with no regularization ($\epsilon = 0.0$) and the estimated source wavelet (bottom).

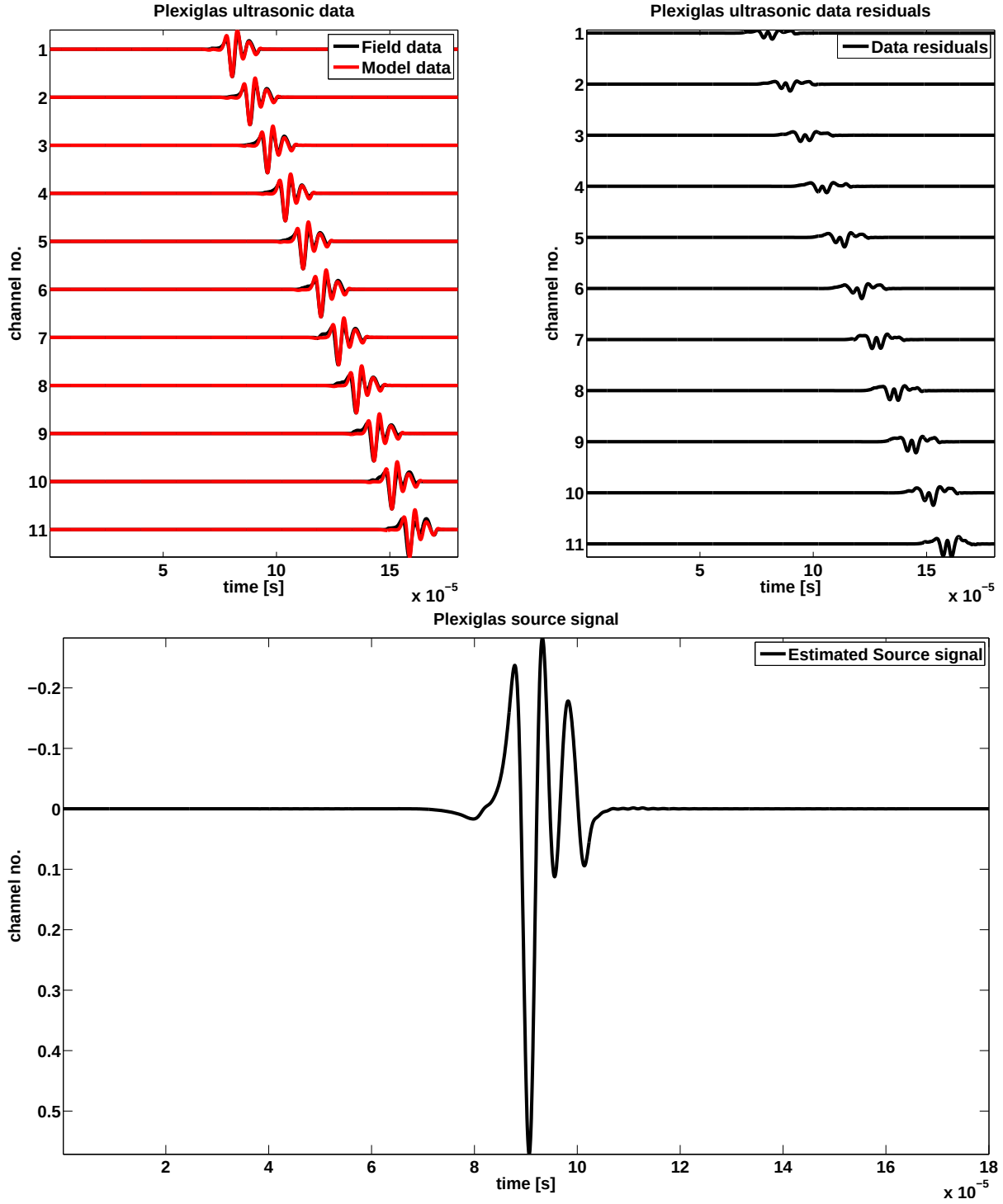


Figure 8.9: Elastic waveform inversion of ultrasonic data - step 8. Resulting seismic sections for the estimated source wavelet (top) with regularization ($\epsilon = 1e1$) and the estimated source wavelet (bottom).

10. Copy wavelet_US_opt.dat to /par/wavelet, edit DENISE_US.inp

```
SOURCE_FILE = ./source/source_wavelet.dat
QUELLART = 3
SIGNAL_FILE = ./wavelet/wavelet_US_opt.dat
```

    set the time delay td in the source-file *source_wavelet.dat* to 0.0 and run DENISE again.

11. Copy DENISE/su/DENISE_US_y.su.shot1.it1 to the FD_seis directory and run the Matlab script Plot_US_Residuals.m. You should get the following seismic sections:

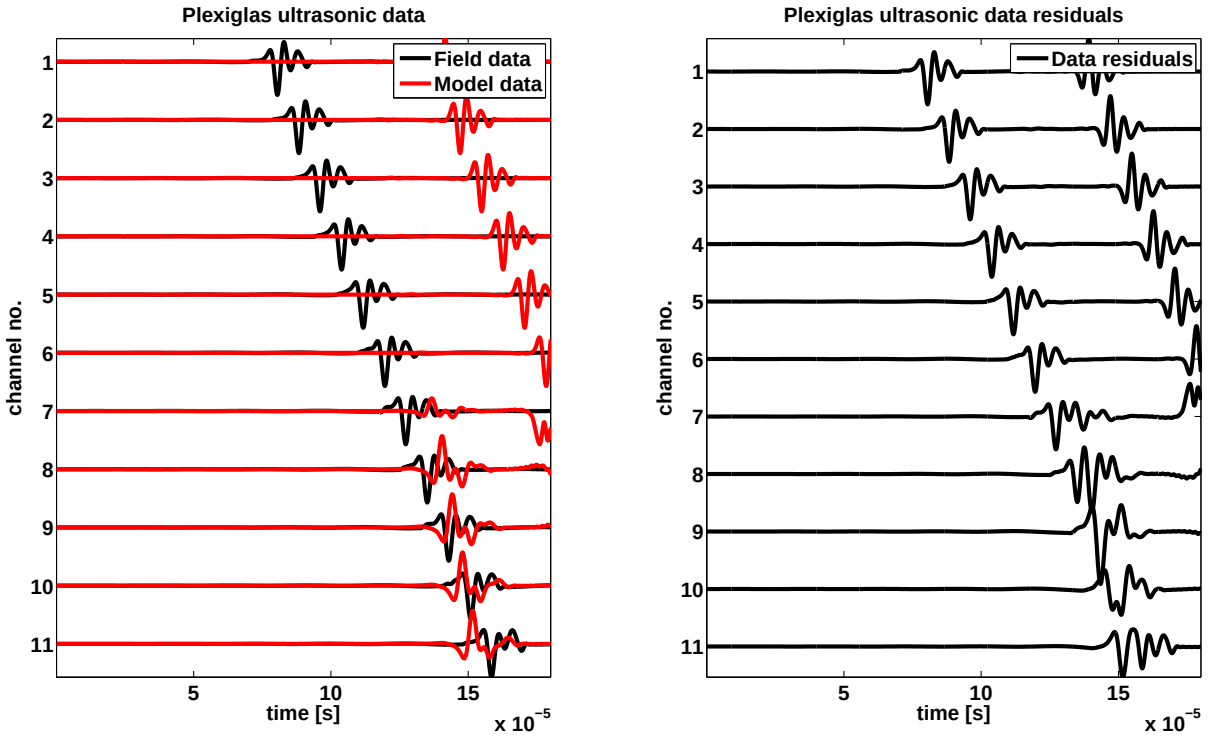


Figure 8.10: Elastic waveform inversion of ultrasonic data - step 9 (model result with estimated source wavelet).

12. Similar to step 6 measure the time delay td in Matlab. In this case the first arrival of the measured data is earlier than the modelled arrival. Therefore the time delay is negative, td=-5.875e-5 s:

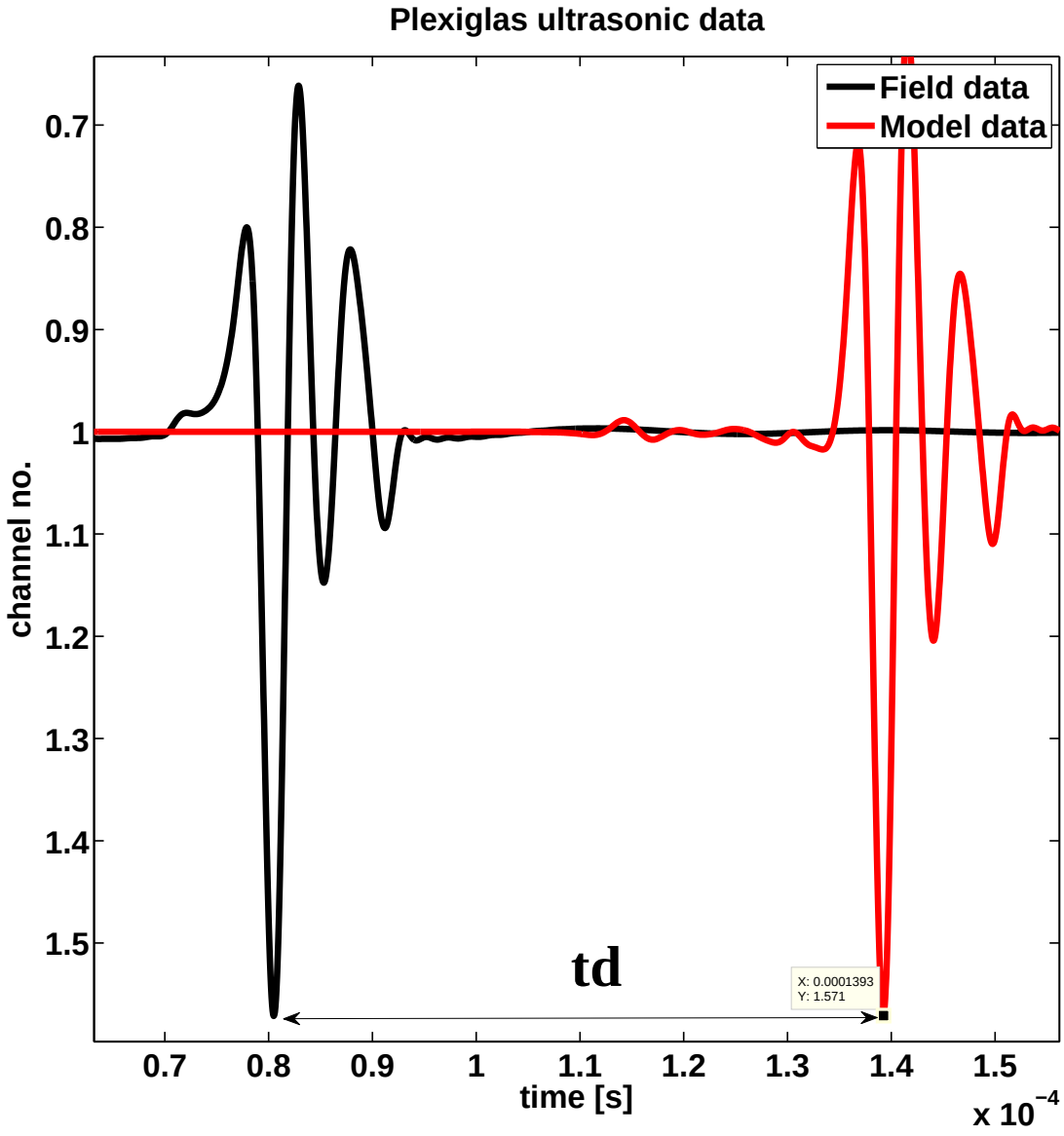


Figure 8.11: Elastic waveform inversion of ultrasonic data - step 10 (measure time delay).

13. Change the time delay td in source_wavelet.dat according to the measured time delay and run DENISE again. Copy the resulting DENISE/su/DENISE_US_y.su.shot1.it1 to the FD_seis directory and run the Matlab script Plot_US_Residuals.m. You should get the following seismic sections with an approximately well fitted first arrival wavelet:

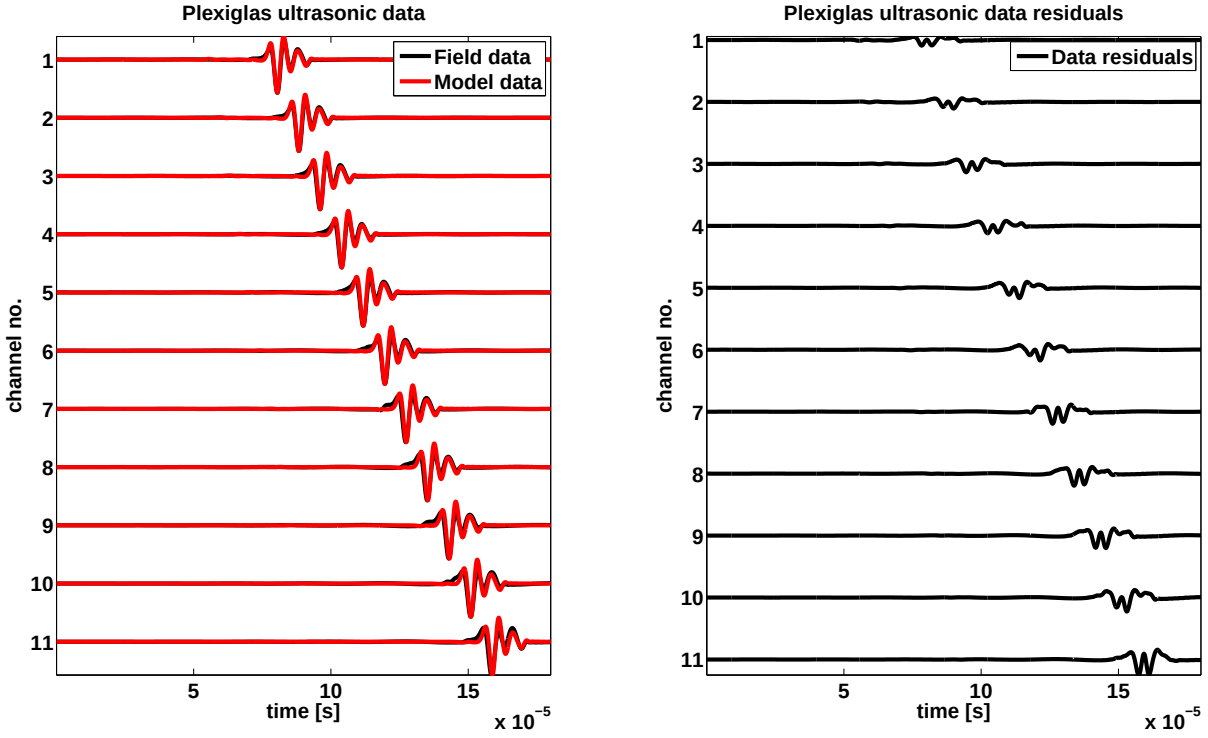


Figure 8.12: Elastic waveform inversion of ultrasonic data - step 11 (model result with estimated source wavelet and corrected time delay).

Chapter 9

Example 4 - shallow seismics

9.1 Inversion of viscoelastic observations

You can find the input files for the small toy example in the directory `par/in_and_out/toy_example`. To run the example you can use the shell script `run_toy_example.sh` in the directory `par`. It is adjusted for a PC with at least 4 CPUs. If you have less CPUs you have to adjust the number of processors in the input files as well as in the call of DENISE in the shell script. The shell script includes all relevant steps. First all libraries and DENISE are compiled. (Do not get nervous about the huge output during compiling.) If you run into problems during this step you maybe have to adjust the variables in `Makefile_var` in the directory `contrib`. Afterwards DENISE starts to simulate observed data for the inversion. The simulated seismograms are renamed for the inversion and DENISE is again compiled with another model function which creates the initial models for the inverison on the fly (see section 6.6). The last step in the shell script is the call of DENISE to start the inversion.

The true model used for the simulation of the observed data is shown in Figure 9.1 whereat the shot positions are marked by the red stars and the CPML frame is marked by the black dashed line. We consider a viscoelastic medium in this test and approximate a constant quality factor of $Q_s = Q_p = 20$ in the analyzed frequency band up to 70 Hz with three relaxation mechanisms of a generalized standard linear solid. The 36 two component receivers used in the inversion are located equidistantly between the sources with a receiver spacing of 1 m.

In the inversion we only inverted for the shear wave velocity model. The initial model is a linear gradient and is shown in Figure 9.2. For the P wave velocity model as well as for the density model the true models (see Figure 9.1(b) and (c)) are used. Furthermore, we use the correct rheological model.

You will need less than 1 GB working memory to run the inversion and approximately 250 MB of data are written to hard disk. Normally 137 iteration steps are calculated and the inversion will take approximately 6 hours. The inversion results for the shear wave velocity are shown in Figure 9.3. Figure 9.3(a) shows an image plot and Figure 9.3(b) shows vertical velocity profiles through the shear wave velocity models (thick grey line: true model; black dashed line: initial model; blue line: inverted model at $x = 20$ m; red dashed line: inverted model at $x = 25$ m).

You can use the script `mfiles/Plot_results_toy_example.m` to visualize the results of the toy example. This script can be used with Matlab or Octave.

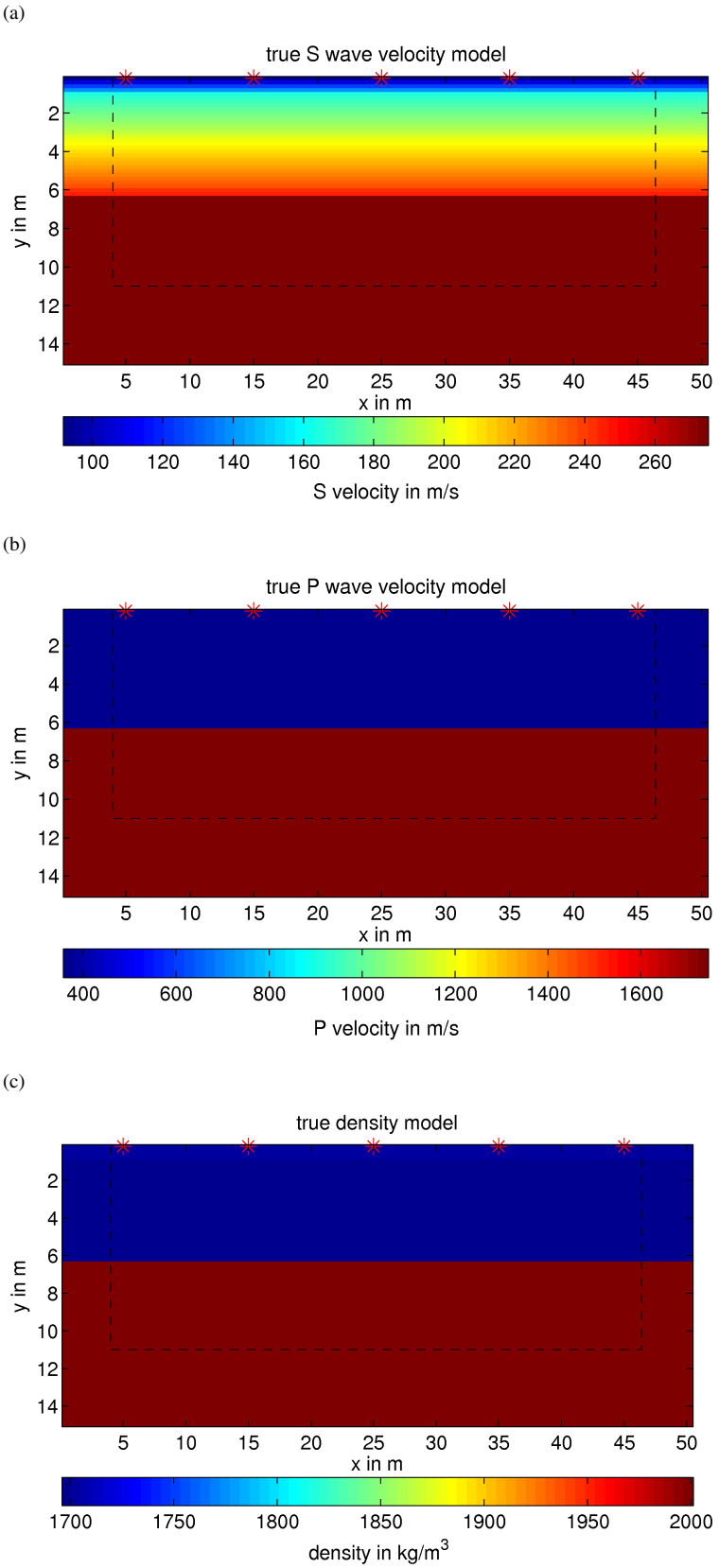


Figure 9.1: True models for (a) the S wave velocity, (b) the P wave velocity and (c) the density used for the calculation of observed data. The red stars mark the five shots used in the inversion. The CPML frame is marked by the black dashed line.

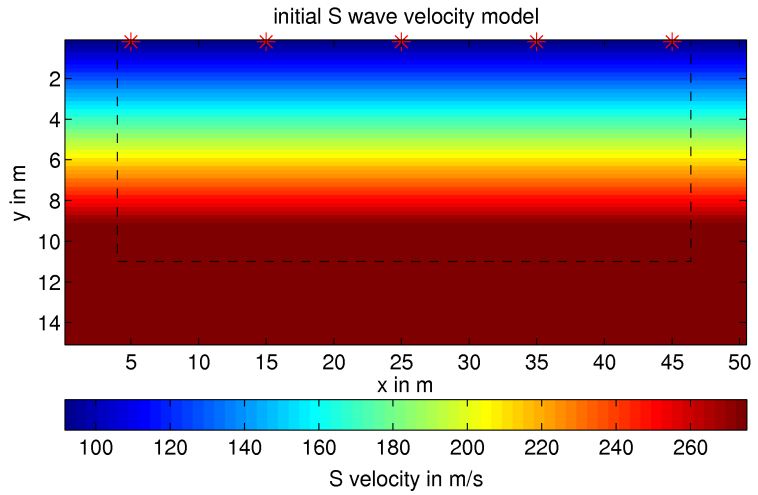


Figure 9.2: Initial shear wave velocity model. The red stars mark the five shots used in the inversion. The CPML frame is marked by the black dashed line.

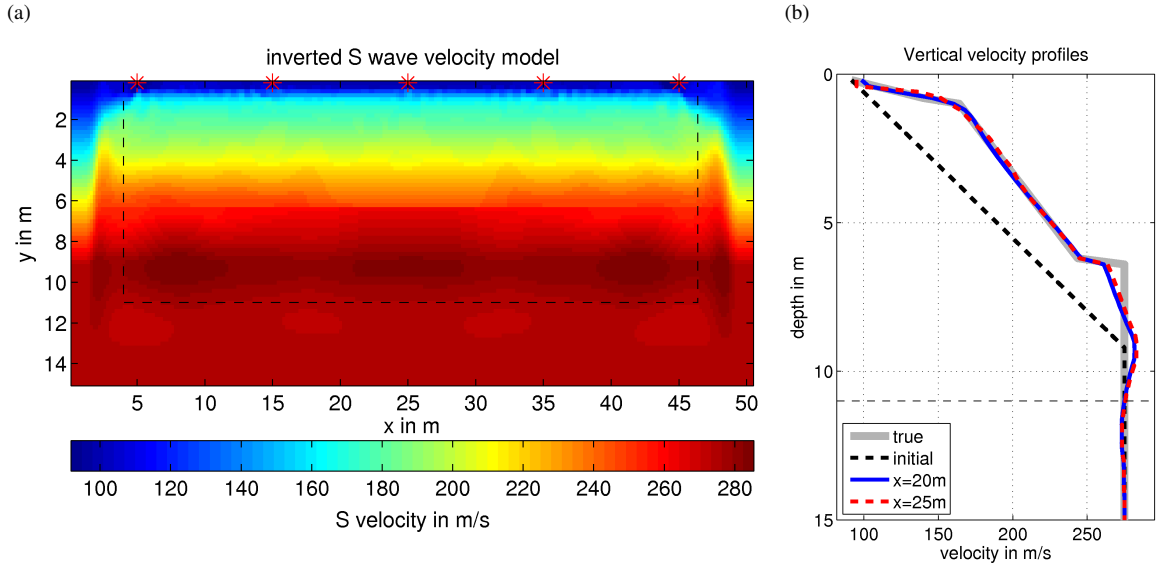


Figure 9.3: Inverted shear wave velocity model. (a) shows an imageplot of the inversion result. The red stars mark the source positions and the black dashed line marks the CPML frame. (b) shows vertical velocity profiles of the shear wave velocity models. The true model is plotted with the thick grey line, the initial velocity model is represented by the dashed black line and two vertical profiles at x=20 m and x=25 m of the inverted model are plotted in red and blue. The CPML frame is again marked by the thin black line at 11 m depth.

Bibliography

H. Ben-Hadj Ali, S. Operto, and J. Virieux. Velocity model-building by 3D frequency-domain full-waveform inversion of wide-aperture seismic data. *Geophysics*, 73(5):101–117, 2008.

J. Blanch, J. Robertsson, and W. Symes. Modeling of a constant Q: Methodology and algorithm for an efficient and optimally inexpensive viscoelastic technique. *Geophysics*, 60(1):176–184, 1995.

T. Bohlen. Parallel 3-D viscoelastic finite-difference seismic modelling. *Computers & Geosciences*, 28(8):887–899, 2002.

T. Bohlen. *Interpretation of Measured Seismograms by Means of Viscoelastic Finite Difference Modelling*. PhD thesis, Kiel University, 1998.

T. Bohlen and E. Saenger. Accuracy of heterogeneous staggered-grid finite-difference modeling of Rayleigh waves. *Geophysics*, 71(4):T109–T115, 2006.

A. Brenders and R. Pratt. Full waveform tomography for lithospheric imaging: results from a blind test in a realistic crustal model. *Geophys. J. Int.*, 168:133–151, 2007.

R. Brossier. *Imagerie sismique á deux dimensions des milieux visco-élastiques par inversion des formes d'ondes : développements méthodologiques et applications*. PhD thesis, Universite de Nice - Sophia Antipolis, 2009.

Y. Choi and T. Alkhalifah. Application of multi-source waveform inversion to marine streamer data using the global correlation. *Geophys. Prosp.*, 60:748–758, 2012.

Y. Choi, D. Min, and C. Shin. Frequency-domain elastic full waveform inversion using the new pseudo-hessian matrix: Experience of elastic marmousi-2 synthetic data. *Bull., Seis Soc. Am.*, 98:2402–2415, 2008a.

Y. Choi, D. Min, and C. Shin. Two-dimensional waveform inversion of multi-component data in acoustic-elastic coupled media. *Geophysical Prospecting*, 56:863–881, 2008b.

R. Courant, K. Friedrichs, and H. Lewy. Über die partiellen Differenzengleichungen der mathematischen Physik. *Mathematische Annalen*, 100:32–74, 1928.

R. Courant, K. Friedrichs, and H. Lewy. On the partial difference equations of mathematical physics. *IBM Journal*, pages 215–234, March 1967.

M. Dougherty and R. Stephen. Seismic energy partitioning and scattering in laterally heterogeneous ocean crust. *Pure Appl. Geophys.*, 128(1/2):195 – 239, 1988.

T. Forbriger. *Inversion flachseismischer Wellenfeldspektren*. PhD thesis, Stuttgart University (http://elib.uni-stuttgart.de/opus/volltexte/2001/861), 2001.

O. Gauthier, J. Virieux, and A. Tarantola. Two-dimensional nonlinear inversion of seismic waveforms - numerical results. *Geophysics*, 51(7):1387–1403, 1986.

O. Holberg. Computational aspects of the choice of operator and sampling interval for numerical differentiation in lage-scale simulation of wave phenomena. *Geophysical Prospecting*, 35:629–655, 1987.

C. Jastram. *Seismische Modellierung mit Finiten Differenzen höherer Ordnung auf einem Gitter mit vertikal variierendem Gitterabstand*. PhD thesis, Universität Hamburg, 1992.

D. Köhn. *Time Domain 2D Elastic Full Waveform Tomography*. PhD thesis, Kiel University, 2011.

D. Komatitsch and R. Martin. An unsplit convolutional perfectly matched layer improved at grazing incidence for the seismic wave equation. *Geophysics*, 72(5):155 – 167, 2007.

A. Kurzmann, D. Köhn, A. Przebindowska, N. Ngyuen, and T. Bohlen. Performance of acoustic full waveform tomography for different acquistion geometries. In *12th annunal report of the Wave Inversion Technology consortium*, pages 117–125, 2008.

A. Levander. Fourth-order finite-difference P-SV seismograms. *Geophysics*, 53(11):1425–1436, 1988.

G. Martin, R. Wiley, and K. Marfurt. Marmousi2 - An elastic upgrade for Marmousi. *The Leading Edge*, 25:156–166, 2006.

R. Martin and D. Komatitsch. An unsplit convolutional perfectly matched layer technique improved at grazing incidence for the viscoelastic wave equation. *Geophys. Prosp.*, 179(1):333–344, 2009.

P. Moczo, J. Kristek, and L. Halada. *The Finite-Difference Method for Seismologists. An Introduction*. Comenius University, Bratislava, 2004.

P. Mora. Nonlinear two-dimensional elastic inversion of multioffset seismic data. *Geophysics*, 52:1211 – 1228, 1987.

P. Morse and H. Feshbach. *Methods of theoretical physics*. McGraw-Hill Book Company, New York, 1953.

J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 1999.

A. Pica, J. Diet, and A. Tarantola. Nonlinear inversion of seismic reflection data in a laterally invariant medium. *Geophysics*, 55(3):284–282, 1990.

R. Pratt. Velocity models from frequency-domain waveform tomography: Past, present and future. In *66th EAGE conference and exhibition, Expanded Abstracts*, pages 181–182, Paris, France, 2004.

R. Pratt. Inverse theory applied to multi-source cross-hole tomography. Part II: Elastic wave-equation method. *Geophysical Prospecting*, 38:311–329, 1990.

R. Pratt. Seismic waveform inversion in the frequency domain, Part 1: Theory and verification in a physical scale model. *Geophysics*, 64:888–901, 1999.

R. Pratt and M. Worthington. Inverse theory applied to multi-source cross-hole tomography. Part I: Acoustic wave equation method. *Geophysical Prospecting*, 38:287–310, 1990.

R. Pratt, F. Gao, C. Zelt, and A. Levander. The limits and complementary nature of traveltime and waveform tomography. In *International Conference of Sub-basalt imaging, Expanded Abstracts*, pages 181–182, Cambridge, England, 2002.

R. Pratt, L. Huang, N. Duric, and P. Littrup. Sound-speed and attenuation imaging of breast tissue using waveform tomography of transmission ultrasound data. 2007.

J. Robertsson, J. Blanch, and W. Symes. Viscoelastic finite-difference modeling. *Geophysics*, 59(9):1444–1456, 1994.

J. Robertsson, A. Levander, W. Symes, and K. Holliger. A comparative study of free-surface boundary conditions for finite-difference simulation of elastic/viscoelastic wave propagation. pages 1277–1280, Houston, Texas, 1995.

H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *The Computer Journal*, 3: 175–184, 1960.

D. Sheen, K. Tuncay, C. Baag, and P. Ortoleva. Time domain Gauss-Newton seismic waveform inversion in elastic media. *Geophys. J. Int.*, 167:1373–1384, 2006.

R. Shipp and S. Singh. Two-dimesional full wavefield inversion of wide-aperture marine seismis streamer data. *Geophys. J. Int.*, 151:325–344, 2002.

F. Sourbier, S. Operto, J. Virieux, P. Amestoy, and J. L'Excellent. FWT2D: a massively parallel program for frequency domain full-waveform tomography of wide-aperture seismic data - part 1: algorithm. *Computer & Geosciences*, 35:487–496, 2009a.

F. Sourbier, S. Operto, J. Virieux, P. Amestoy, and J. L'Excellent. FWT2D: a massively parallel program for frequency domain full-waveform tomography of wide-aperture seismic data - part 2: numerical examples and scalability analysis. *Computer & Geosciences*, 35:496–514, 2009b.

A. Tarantola. *Inverse Problem Theory*. SIAM, 2005.

A. Tarantola. Inversion of seismic reflection data in the acoustic approximation. *Geophysics*, 49:1259–1266, 1984a.

A. Tarantola. Linearized inversion of seismic reflection data. *Geophysical Prospecting*, 32:998–1015, 1984b.

A. Tarantola. A strategy for nonlinear elastic inversion of seismic reflection data. *Geophysics*, 51:1893–1903, 1986.

A. Tarantola. Theoretical background for the inversion of seismic waveforms, including elasticity and attenuation. *PAGEOPH*, 128:365–399, 1988.

R. Versteeg. The marmousi experience: Velocity model determination on a complex data set. *The Leading Edge*, 13: 927–936, 1994.

J. Virieux. P-SV wave propagation in heterogeneous media: velocity-stress finite-difference method. *Geophysics*, 51 (4):889–901, 1986.

A. Zerwer, M. Polak, and J. Sanatamarina. Wave propagation in thin plexiglas plates: implications for rayleigh waves. *NDT & E International*, 33:33–41, 2000.